M20 PERSONAL COMPUTER

Linguaggio BASIC

Manuale generale

olivetti

M20 PERSONAL COMPUTER

Linguaggio BASIC

Manuale generale

olivetti

PREFAZIONE

Questo manuale descrive, in modo semplice, l'uso del linguaggio BASIC del Sistema M20 dell'OLIVETTI. La descrizione si avvale di molte figure, tabelle ed esempi, inoltre le istruzioni e i comandi che vengono usati insieme o hanno una funzione simile, vengono illustrati nello stesso capitolo.

Non è indispensabile una precedente esperienza di programmazione, ma è richiesta una conoscenza dei concetti di base sulla elaborazione dei dati.

L'APPENDICE E PRESENTA LE ISTRUZIONI, I COMANDI E LE FUNZIONI BASIC IN ORDINE ALFABETICO CON RELATIVO RIFERIMENTO AL NUMERO DI PAGINA, PER PERMETTERE UNA RAPIDA CONSULTAZIONE.

Questo manuale ha validità per tutti i modelli dell'M20.

RIFERIMENTI:

- L1 M20
  Professional Computer Operating System (PCOS) - Guida Utente
  Code 3985290 E

DISTRIBUZIONE: Generale (G)

TERZA EDIZIONE: Febbraio 1983

PRIMO AGGIORNAMENTO: Giugno 1983

RELEASE: 1.3 e successive

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis-10015 IVREA (Italy)

INDICE

INDICE

1. COSA È IL BASIC?

SOMMARIO

Questo capitolo illustra le principaTi caratteristiche del Iinguaggio BASIC deT Modello 20 (M20).

Dopo un breve cenno ai due ambienti operativi presenti su questo sistema, iI PCOS (Professional Computer Operating System) e il BASIC, si spiega come usare la tastiera, come eseguire caTcoli immediati, come impostare ed eseguire un programma BASIC e quali sono le modalità operative in BASIC.

INOICE

IL LINGUAGGIO BASIC

IL BASIC (Beginner's All-purpose Symbolic Instruction Code) è un linguaggio di alto livello e di uso generale (general purpose).

POSSIAMO USARE IL BASIC PER RISOLVERE SIA PROBLEMI COMMERCIALI CHE SCIENTIFICI.

IL BASIC E' UN INSIEME DI ISTRUZIONI E COMANDI DI IMMEDIATA INTERPRETAZIONE. RISULTA QUINDI FACILE IMPARARLO E USARLO.

Diverse versioni del linguaggio BASIC sono disponibili su diversi sistemi. La prima versione del BASIC fu sviluppata al Dartmouth College da John G. Kemeny e Thomas E. Kurts. D'ora in poi, parlando di BASIC, faremo riferimento alla versione implementata sul Modello 20 (M20).

IL BASIC M20 E' UNA VERSIONE DEL BASIC MICROSOFT ESTESA CON PRESTAZIONI GRAFICHE E LA GESTIONE DELL'INTERFACCIA STANDARD IEEE 488.

AMBIENTI PCOS E BASIC

Il sistema M20 può essere semplicemente definito come un computer e un insieme di programmi forniti dalla Olivetti e residenti su un floppy disk da 5¼ in. (disco sistema). Questi programmi possono essere anche residenti su hard disk per i sistemi M20 con questa unità.

Questi vengono detti "Programmi di Sistema". Questi Programmi di Sistema, che comprendono il PCOS e il BASIC, permettono di dare istruzioni al computer in un linguaggio simile alla lingua inglese. Essi convertono le istruzioni fornite dall'utente in un linguaggio macchina che può così essere interpretato dal computer. L'utente interagisce con il computer usando comandi PCOS e BASIC e insiemi di istruzioni che formano un programma BASIC.

Nota: D'ora in avanti useremo:

- il termine "dischetto" invece di floppy disk da 5¼ in.

- il termine "disco" per specificare sia un dischetto sia l'hard disk.

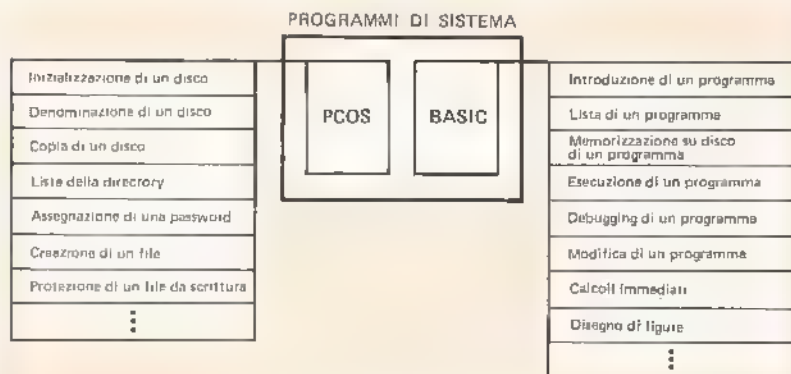



Figura 1-1 Programmi di Sistema

USO DELLA TASTIERA

La tastiera permette di impostare qualsiasi tipo di testo e di introdurre i caratteri di controllo.

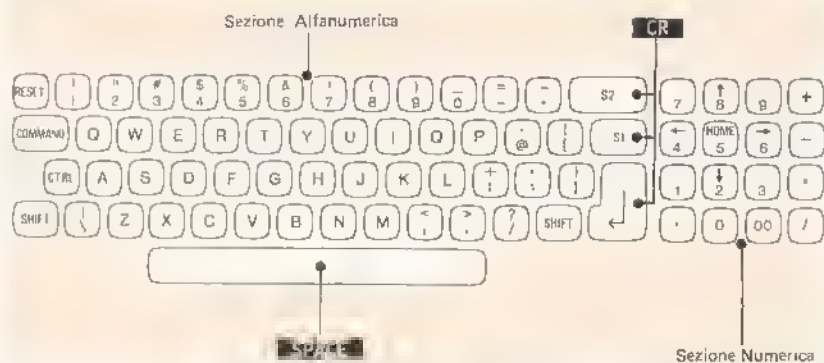



Figura 1-2 La Tastiera (Versione USA-ASCII)

Nota: Tutti i caratteri indicati in questo manuale si riferiscono alla versione USA-ASCII della tastiera. L'Appendice B elenca le varie equivalenze nazionali per quei caratteri ASCII, che appaiono su video o su stampante.

Quando, in questo manuale, si vuole mettere in evidenza quali sono i tasti che l'utente deve premere per realizzare una certa funzione, verrà riportata l'esatta sequenza dei tasti in negativo (caratteri bianchi su fondo nero). In detta sequenza possono figurare anche i tasti indicati in negativo nella figura 1-2. Ad esempio:

L I S T CR

Nota: Per convenzione, useremo il simbolo CR per indicare uno qualsiasi dei tre tasti di ritorno a capo/interlinea (S1 S2 e ↵) e il simbolo SPACE per indicare la barra spaziatrice e SHIFT per indicare uno dei due tasti shift (solo la tastiera USA - ASCII e USA - ASCII con verbi BASIC hanno i due tasti Shift siglati, le altre tastiere hanno questi tasti privi di sigla).

Volendo ricordare all'utente di premere CR per inviare una linea al sistema, aggiungeremo il simbolo CR al termine della linea medesima. Per esempio:

DELETE 100 - 200 CR

COME INTRODURRE I CARATTERI

SE l'utente...	ALLORA...
preme un tasto (o una combinazione di tasti)	il carattere che rappresenta viene subito visualizzato su video. Di volta in volta che l'utente imposta dei caratteri, il cursore (█) lampeggia e indica la posizione che verrà occupata dal carattere successivo
vuole introdurre un carattere minuscolo o il simbolo inferiore (dei tasti con due simboli)	deve semplicemente premere quel tasto, per esempio A per a

vuole introdurre un carattere maiuscolo o il simbolo superiore (dei tasti con due simboli)	deve premere contemporaneamente al tasto uno dei due tasti SHIFT per esempio SHIFT A per A. L'UTENTE PUO' ABILITARE O DISABILITARE IL FISSA MAIUSCOLE PER LE LETTERE (A-Z) PREMENDO COMMAND ? / (vedere Caratteri di Controllo)
vuole introdurre un numero	può usare la sezione numerica o la prima riga di tasti della sezione alfanumerica
tiene premuto un tasto per un tempo superiore al tempo medio di una battuta (mezzo secondo)	il carattere corrispondente viene introdotto più volte finchè il tasto non viene rilasciato
vuole trasmettere una linea al sistema (una linea BASIC, un comando PCOS, o più dati in risposta a una istruzione INPUT o LINE INPUT)	deve chiudere il messaggio premendo CR, che posiziona il cursore all'inizio della linea successiva
vuole spostare il cursore alla linea successiva (prima di raggiungere il margine del video)	deve premere ripetutamente (o mantenere premuto) SPACE

CARATTERI DI CONTROLLO

I caratteri di controllo vengono impostati premendo contemporaneamente CTRL oppure COMMAND e un altro tasto. La seguente tabella riporta tutti i caratteri di controllo dell'M20 con relativi significati.

SE l'utente imposta...	ALLORA...
CTRL C (Break)	interrompe l'esecuzione di un programma. L'M20 ritorna al BASIC (Stato Comandi) e visualizza Ok (se in ambiente BASIC) o ritorna al PCOS (Stato Comandi) e visualizza > (se in ambiente PCOS). Vedere anche: "Come Correggere gli Errori di Impostazione"

CTRL G	il cursore cambia forma e lampeggia e nessun carattere impostato in tastiera appare su video (Hide State) PER RITORNARE A VISUALIZZARE I CARATTERI L'UTENTE DEVE PREMERE NUOVAMENTE CTRL G OPPURE CR
CTRL H (Tasto di ritorno)	l'ultimo carattere introdotto viene cancellato e il cursore viene arretrato di una posizione
CTRL I (Tabulatore)	il cursore si sposta a destra di otto posizioni
CTRL RESET (Reset logico)	la memoria viene azzerata e il PCOS viene di nuovo caricato in memoria da disco
CTRL S	viene sospeso l'output su video. L'OUTPUT PUO' ESSERE RIPRESO IMPOSTANDO UN TASTO QUALSIASI
CTRL RESET (Escape)	si esce dallo Stato di Inserimento Caratteri, pur rimanendo nello Stato di Editor (v. Cap. 3)
COMMAND ?/	abilita il fissa maiuscole per lettere (A-Z). PER DISABILITARE IL FISSAMAIUSCOLE, L'UTENTE DEVE PREMERE DI NUOVO COMMAND ?/

COME CORREGGERE GLI ERRORI DI IMPOSTAZIONE

L'utente può correggere un errore di impostazione sia prima che dopo aver introdotto una linea.

SE l'utente si accorge...	ALLORA...	OPPURE...
di un errore prima di aver completato una linea (cioè prima di premere CR)	può cancellare l'ultimo (o gli ultimi) caratteri impostati premendo una (o più volte) CTRL H e reimpostare il (o i) caratteri errati	può spostare il cursore alla linea successiva premendo CTRL C e impostarla di nuovo dall'inizio

di un errore dopo aver completato la linea (cioè dopo aver premuto **CR**) E SE si tratta di una linea di programma	può sostituire la linea errata, introducendo una nuova linea con lo stesso numero di linea; LA NUOVA LINEA SOSTITUIRA' IN MEMORIA LA PRECEDENTE	può entrare in Stato Editor e usare i comandi dell'Editor per modificare la linea (vedere Capitolo 3)

CALCOLI IMMEDIATI CON L'M20

L'utente può utilizzare il sistema M20 come una macchina calcolatrice sia quando vuole sapere subito il risultato di una espressione, che quando vuole verificare se un dato programma funziona correttamente o meno.

Siamo in BASIC. Il messaggio Ok appare sul video.

L'utente può impostare **P** **R** **I** **N** **T** (oppure più semplicemente il solo tasto **?**, poi una espressione e **CR**.
L'espressione viene valutata e il suo valore viene visualizzato.
L'utente può anche introdurre **L** **E** **T**, poi il nome di una variabile (cioè una stringa di caratteri il cui primo carattere è una lettera), l'operatore di assegnazione (=), un'espressione e **CR**. L'espressione viene valutata e il suo valore viene memorizzato nella variabile; questa può essere utilizzata in calcoli successivi per rappresentare quel determinato valore.

La seguente tabella riporta alcuni esempi di calcoli immediati.

VIDEO	COMMENTI
PRINT 3 3 Ok	la costante 3 viene visualizzata (possiamo classificare una costante come il caso più semplice di espressione)
PRINT 2+3 5 Ok	L'espressione 2+3 viene valutata e il suo valore (5) viene visualizzato

LET A = 15.21 Ok	la costante 15.21 viene assegnata alla variabile A. L'utente può utilizzare la variabile A in calcoli successivi per rappresentare detto valore.
? A-1 14.21 Ok	l'espressione A-1 viene valutata e il suo valore (14.21) viene visualizzato. Nota: E' possibile impostare '?' invece di PRINT
B = 2.3 Ok	la costante 2.3 viene assegnata alla variabile B. La parola chiave LET può essere omessa, l'utente può iniziare la linea con il nome di una variabile
? A*B 34.983 Ok	l'espressione A*B viene valutata. Il simbolo * significa moltiplicato. Il valore 34.983 viene visualizzato.
? A*B-4Ø -5.Ø17 Ok	l'espressione A*B-4Ø viene valutata e il suo valore (-5.Ø17) viene visualizzato. Si noti che se il valore di una espressione è negativo il segno meno (-) viene sempre visualizzato, mentre se il suo valore è positivo il segno più (+) viene sostituito da uno spazio

UN SEMPLICE PROGRAMMA

Si può utilizzare il sistema M20 anche per memorizzare ed eseguire un programma BASIC.

Eseguendo un programma, l'utente può risolvere problemi che non è possibile risolvere utilizzando il sistema M20 come una qualsiasi macchina calcolatrice.

Un programma BASIC consiste di un insieme di istruzioni.

Un'istruzione è un'entità elementare di programma che dice al BASIC di svolgere una ben determinata operazione.

E' possibile impostare linee con una o più istruzioni. In questo ultimo caso le istruzioni vengono separate l'una dall'altra dal segno di

punteggiatura due punti (:).

In un programma BASIC ogni linea inizia con un numero intero compreso tra Ø e 65529 (estremi inclusi) e finisce quando l'utente preme il tasto **CR**

Siamo in BASIC. Sul video appare il messaggio OK. L'utente può iniziare a introdurre in memoria da tastiera delle istruzioni. Ad esempio le seguenti:

```
10 REM RETTANGOLO1 CR
20 INPUT "Base"; B CR
30 IF B<=0 THEN 20 CR
40 INPUT "Altezza"; H CR
50 IF H<=0 THEN 40 CR
60 LET AREA = B*H CR
70 PRINT "Area=";AREA;" B=";B;" H=";H CR
80 GOTO 20 CR
90 END CR
```

Queste istruzioni formano un programma completo. Questo programma permette di risolvere un problema molto semplice; il problema consiste nel trovare l'area di un rettangolo introducendo da tastiera i valori della base e dell'altezza.

Questo esempio è stato scelto sia per la sua semplicità sia per illustrare un certo numero di prestazioni del linguaggio BASIC.

Non si esclude che si possano realizzare programmi più efficienti per risolvere lo stesso problema (vedere Capitolo 3).

Nel programma appena esaminato abbiamo impostato un'istruzione per ogni linea, ma avremmo anche potuto impostarne più di una, (utilizzando il separatore due punti (:)), riducendo così il numero di linee. Ad esempio:

```
10 REM RETTANGOLO1 CR
20 INPUT "Base";B:IF B<=0 THEN 20 CR
30 INPUT "Altezza";H:IF H<=0 THEN 30 CR
40 LET AREA=B*H CR
50 PRINT "Area=";AREA;" B=";B;" H=";H CR
60 GOTO 20:END CR
```

L'utente può introdurre fino a 255 caratteri per ogni linea (logica), compreso il numero di linea. Una linea logica può comprendere più linee fisiche. Per esempio:

```
20 INPUT "Base";B:IF B<=0
                THEN 20 CR
```

è una linea logica comprendente due linee fisiche. Per cambiare linea prima di raggiungere il margine del video, l'utente deve premere una o più volte SPACE.

PARDLE CHIAVE (KEYWORDS)

Ogni istruzione incomincia con una parola chiave, che è una parola riservata. Una parola chiave è una parola mnemonica della lingua inglese. Dal punto di vista sintattico, deve essere preceduta e seguita da almeno uno spazio.

Nota: Una parola chiave non può essere usata come nome di una variabile.

La parola chiave indica il tipo d'istruzione da eseguire. Dopo la parola chiave l'utente può introdurre uno o più operandi (costanti o variabili) o espressioni. Alcune istruzioni hanno più di una parola chiave ad esempio IF... THEN. Le istruzioni del nostro programma contengono le parole chiave REM, INPUT, IF... THEN, LET, PRINT, GOTO e END. L'utente può introdurre le parole chiave sia in lettere maiuscole che in lettere minuscole. In quest'ultimo caso vengono convertite in lettere maiuscole quando si lista il programma (vedere Capitolo 2).

In BASIC sono parole riservate sia le parole chiave sia i nomi dei comandi (per esempio RUN, LIST, ecc...) e i nomi di funzione (per esempio SIN, COS, ecc...). Vedere Appendici C,D e E per una lista completa.

COSTANTI

I numeri scritti in un programma BASIC (quali ad esempio 0, 150,-31.7) vengono denominati "costanti numeriche" e le stringhe di caratteri (quali "Base", "Altezza", "Area=", " B=" e " H=") vengono denominate "costanti stringa". Questo significa che i loro valori rimangono gli stessi durante tutto il programma. Per esempio, quando la costante 150 appare in un programma, questo valore resta immutato per tutta la durata del programma.

Le costanti numeriche possono essere numeri interi (ad es; 150) o meno (numeri "reali"), ad es. -31.7.

Le costanti stringa sono sempre comprese fra virgolette, a meno che non venga specificato altrimenti. Stringhe non comprese tra virgolette sono ammesse ad esempio all'interno di istruzioni DATA o in risposta a una istruzione INPUT o LINE INPUT.
Per ulteriori dettagli vedere Capitolo 4.

VARIABILI

In un programma se un dato viene rappresentato da un nome, il suo valore può variare durante l'esecuzione. Si parla in questo caso di una "variabile" anzichè di una "costante". Il nome di una variabile non può essere più lungo di 40 caratteri. Il primo carattere deve essere una lettera. Esempi di variabili nel nostro programma sono:
B , H , AREA

Come abbiamo visto per le parole chiave, i nomi delle variabili possono essere introdotti sia in lettere maiuscole che minuscole. In quest'ultimo caso vengono convertite in lettere maiuscole quando si lista il programma.
Una variabile può essere una "variabile semplice" (ad es. le predette variabili L, W, AREA) oppure una "variabile con indice".

Una varibile con indice è un elemento di matrice ("array"), cioè un elemento di una collezione di variabili che hanno tutte lo stesso nome. I vari elementi della matrice si distinguono tramite uno (o più) indici scritti tra parentesi tonde subito dopo il nome comune. Per esempio, se A è il nome di una matrice a una dimensione, A (Ø) è il suo primo elemento, A(1) il secondo ecc..
Una matrice può avere un numero qualsiasi di dimensioni. Una matrice a una dimensione può essere considerata come una lista di dati. Una matrice a due dimensioni può essere considerata una tabella di dati. In questo caso il primo indice individua la riga della tabella e il secondo la colonna. Per esempio B(1,2) rappresenta il valore appartenente alla seconda riga e alla terza colonna.
Per ulteriori informazioni vedasi il Capitolo 4.

FUNZIONI

Possiamo classificare le funzioni come funzioni di sistema ("built-in functions") o definite dall'utente ("user-defined functions").
Parleremo brevemente qui di funzioni di sistema, mentre le funzioni definite dall'utente verranno illustrate nel seguito (vedasi Capitolo 9 dove si possono anche trovare ulteriori dettagli sulle funzioni di sistema).

Le funzioni di sistema consentono di eseguire molte tra le più comuni funzioni matematiche, quali la radice quadrata, il seno o il logaritmo o funzioni per l'elaborazione delle stringhe (quali l'estrazione di una sottostringa). E' sufficiente scrivere il nome di una di queste funzioni in un'istruzione, perché la funzione venga eseguita. Al nome della funzione può seguire, in parentesi tonde, il valore di uno o più argomenti (per esempio SIN(1.5)).

Una funzione può essere un operando nell'ambito di una espressione (ad es. 1+2*COS(.4)) e gli argomenti possono anche essere delle espressioni (ad es. LOG(45/7)).

Per esempio:

SIN(1,5)	è .997495	(SIN calcola il seno dell'argomento)
1+2*COS(.4)	è 2.84212	(COS calcola il coseno dell'argomento)
LOG(45/7)	è 1.86Ø75	(LOG calcola il logaritmo naturale dell'argomento)
SQR(1Ø)	è 3.16228	(SQR calcola la radice quadrata dell'argomento)

ESPRESSIONI

Possiamo classificare le espressioni in 4 categorie:

- espressioni numeriche
- espressioni stringa
- espressioni di confronto
- espressioni logiche

Parleremo brevemente qui dei primi tre tipi di espressioni, mentre le espressioni logiche verranno definite più avanti (vedi Capitolo 6, dove descriveremo anche gli altri tipi di espressioni più in dettaglio).

Espressioni numeriche

Un'espressione numerica può essere una costante numerica, una variabile numerica semplice, una variabile numerica con indice, una funzione numerica o una successione di questi elementi legati tramite simboli speciali, detti "operatori numerici".

Gli operatori numerici sono:

+ addizione (es. A+B+C)

- Sottrazione (ES. A-B)

\ divisione tra interi (gli operandi vengono arrotondati all'intero più vicino prima che la divisione venga eseguita e il quoziente viene troncato a valore intero, per es. 25.68 \ 6.99 è 3)

MOD divisione modulo (fornisce quel numero intero che è il resto di una divisione tra interi, per es. 25.68 MOD 6.99 è 5, dato che 26/7 è 3 con resto 5)

* moltiplicazione (es. A*B)

/ divisione (es. A/B)

- cambiamento di segno (es. -A è 35, se A vale - 35)

∧ elevamento a potenza (es. A∧B)

Espressioni stringa

Un'espressione stringa può essere una costante stringa, una variabile semplice stringa, un variabile stringa con indice, una funzione stringa o una successione di questi elementi concatenati mediante il segno di addizione (+).

Utilizzando questo segno le stringhe possono essere "concatenate", cioè al primo valore stringa viene aggiunto in coda il secondo e così via fino a ottenere un'unica stringa risultante.

Per esempio:

```
10 A$ = "Chicago,"
20 B$ = "IL.,"
30 C$ = A$+B$+"USA"
```

La concatenazione nell'istruzione 30 farà sì che venga assegnato alla variabile stringa C$ il valore:

```
Chicago,IL.,USA
```

Epressioni di confronto

Dette espressioni confrontano due espressioni numeriche oppure due espressioni stringa per mezzo di un operatore di confronto.

Gli operatori di confronto sono:

=	uguale a
>	maggiore di
<	minore di
>= o =>	maggiore di o uguale a
<= o =<	minore di o uguale a
<>o ><	diverso da

Il risultato di un'espressione di confronto può essere vero o falso e può essere usato per decidere dove passare il controllo dell'esecuzione di un programma.

Per esempio è stata usata un'espressione di confronto nell'istruzione:

```
3Ø IF L<=Ø THEN 2Ø
```

Questa istruzione fa ritornare il controllo all'istruzione 2Ø se il valore della variabile L è negativo o uguale a zero.

USO DEGLI SPAZI

L'utente, per rendere il suo programma più leggibile, può introdurre degli spazi in qualsiasi posizione. Infatti non ci sono limitazioni nell'uso degli spazi, ad eccezione delle seguenti:

- almeno uno spazio deve precedere e seguire una parola chiave,

- gli spazi sono significativi solo all'interno delle costanti stringa,

- gli spazi non sono ammessi all'interno delle costanti numeriche (inclusi i numeri di linea), le parole chiave, i nomi delle variabili e i nomi delle funzioni.

Per esempio:

```
2Ø INPUT "Base";B
e
2Ø   INPUT   "Base"; B
```

sono istruzioni equivalenti, ma

```
20 INPUT "B a s e";B
```

è una istruzione diversa, dato che contiene una stringa di caratteri più lunga.

COMMENTI

L'utente può documentare il proprio programma tramite l'istruzione REM (Remark). Dopo la parola chiave REM è possibile scrivere una qualsiasi stringa di caratteri stampabili ASCII. Per esempio:

```
10 REM RETTANGOLO1 CR
```

E' anche possibile inserire commenti in un programma introducendo una stringa di caratteri stampabili ASCII preceduta da un apostrofo (') e chiusa da CR
Questa stringa è detta "campo commento".

Per esempio:

```
80 GOTO 100 'Cicla indefinitamente CR
```

Sia le istruzioni REM, sia i campi commento possono essere inseriti in un punto qualsiasi di un programma. Dal punto di vista dell'esecuzione due programmi uno con, l'altro senza commenti sono assolutamente equivalenti, ma i commenti figurano nel listing (lista) del programma e consentono di avere un programma di facile comprensione in quanto "autodocumentato". Per ulteriori dettagli vedere Capitolo 2.

ESECUZIONE DEL PROGRAMMA

Proviamo a eseguire il programma che calcola l'area di un rettangolo. Se le linee del programma sono state impostate in tastiera una dopo l'altra (e l'M20 non è stato spento nel frattempo) il programma è in memoria. Impostiamo: L I S T CR; il listing del programma apparirà su video. Alla fine del listing appare Ok.
Impostiamo allora R U N CR.

VIDEO

```
LIST
1Ø REM RETTANGOLO1
2Ø INPUT "Base";B
3Ø IF B< = Ø THEN 2Ø
4Ø INPUT "Altezza";H
5Ø IF H<=Ø THEN 4Ø
6Ø LET AREA=B*H
7Ø PRINT "Area=";AREA;" B=";B;" H=";H
8Ø GOTO 2Ø
9Ø END
Ok
RUN
Base? 3.5
Altezza? 4.2
Area= 14.7   B= 3.5   H= 4.2
Base? -7.3
Base?  7.3
Altezza? 1.3Q
?Redo from start
Altezza? 1.32
Area= 9.636   B= 7.3   H= 1.32
Base?^C
Break in 2Ø
Ok
```

COMMENTI

l'M20 inizia a eseguire le istruzioni in modo sequenziale a partire dall'istruzione 1Ø (che in questo caso non viene eseguita dato che è un commento).

Quando il controllo raggiunge una istruzione INPUT (vedere le istruzioni 2Ø e 4Ø) l'esecuzione del programma viene sospesa e l'M20 emette un messaggio indicando all'utente che deve introdurre un valore da tastiera. Ad esempio possiamo impostare 3.5 per la base e 4.2 per l'altezza.

L'istruzione 6Ø calcola il valore della variabile AREA. L'istruzione 7Ø visualizza questo valore, come pure i valori delle variabili B e H. L'istruzione 8Ø riporta il controllo all'istruzione 2Ø.

Se l'utente introduce un valore negativo per B (ad es.-7.3), l'istruzione 2Ø viene eseguita di nuovo, dato che l'istruzione 3Ø ritorna il controllo all'istruzione 2Ø se il valore di B è negativo o zero.

Se l'utente introduce un valore negativo di H, l'istruzione 4Ø viene eseguita di nuovo, dato che l'istruzione 5Ø ritorna il controllo all'istruzione 4Ø, se il valore di H è negativo o zero.

Se l'utente introduce un valore stringa per B o H (ad es.1.3Q per H) l'M20 visualizza un messaggio d'errore:

?Redo from start

e l'utente deve introdurre un nuovo valore. L'esecuzione di questo programma termina solo se l'utente preme **CTRL** **C**. In tal caso l'M20 visualizza un messaggio di interruzione (Break) e passa in Stato Comandi. Per riprendere l'esecuzione l'utente deve impostare:
C **O** **N** **T** **CR**.

MODALITA' OPERATIVE

Il BASIC ha tre modalità operative.

Figura 1-3 Modalità Operative

STATO COMANDI

Quando l'M20 entra in Stato Comandi, visualizza il messaggio:

Ok

In Stato Comandi, il BASIC non accetta ciò che l'utente imposta in tastiera finchè l'utente non preme **CR**.

Linee Programma e Linee Immediate

L'M20 non prende in considerazione eventuali spazi iniziali su una linea BASIC, ma interpreta il primo carattere diverso da uno spazio. Se questo carattere non è una cifra, il BASIC considera la linea come una "linea immediata" (cioè ad esecuzione immediata). Se è una cifra, il BASIC considera la linea come una "linea di programma" (vedere più avanti).

SE...	ALLORA...
l'utente imposta una linea di programma, cioè un numero di linea (tra Ø e 65529), seguito da uno o più comandi o istruzioni BASIC separati da due punti (:) e alla fine preme CR	la linea viene trasferita in memoria, quando l'utente preme CR. La linea non viene eseguita finchè l'utente non imposta R U N CR. Per esempio: 1ØØ PRINT "Il LOG di 5 è";LOG(5) è una "linea di programma". Quando l'utente preme CR, il BASIC trasferisce in memoria la linea. Per eseguire la linea si deve impostare R U N CR
l'utente imposta una linea immediata, cioè uno o più comandi o istruzioni BASIC separati da due punti (:) e alla fine preme CR	la linea viene eseguita non appena l'utente preme CR. PRINT "Il LOG di 5 è";LOG(5) è una "linea immediata". Quando l'utente preme CR, il BASIC la esegue.
l'utente imposta una sequenza di linee di programma.	le linee vengono trasferite in memoria in modo da formare un programma BASIC. Queste linee vengono memorizzate secondo la sequenza dei numeri di linea, indipendentemente dall'ordine in cui sono state introdotte. Il programma non viene eseguito finchè l'utente non imposta R U N CR

Modi

Lo Stato Comandi comprende i seguenti Modi:

- Modo Immediato (o Diretto), quando l'utente imposta una linea ad esecuzione immediata

- Modo Programma, quando l'utente imposta una linea di programma.

STATO ESECUZIONE

Si dice che il sistema M20 è in Stato Esecuzione, quando esegue sia istruzioni o comandi BASIC (Stato Esecuzione BASIC), sia comandi PCOS (Stato Esecuzione PCOS).
Un programma BASIC viene eseguito secondo la sequenza ascendente dei numeri di linea, a meno che un'istruzione di controllo come GOTO, ON...GOTO, IF...THEN...ELSE, IF...GOTO...ELSE, FOR/NEXT, WHILE/WEND specifichi altrimenti.

STATO EDITOR DI LINEA

Il sistema BASIC comprende un Editor per poter correggere le linee di programma. Questo è molto utile per poter modificare linee lunghe e complesse senza bisogno di riscriverle completamente.

SE l'utente desidera modificare...	ALLORA egli deve impostare...	il BASIC visualizza
la linea attuale (numero di linea ll...l)	E D I T SPACE . CR	ll...l
una linea specificata (numero di linea nn...n)	E D I T SPACE n n ... n CR	nn...n

Nota: La linea attuale è l'ultima linea impostata, modificata o che ha dato luogo a un messaggio d'errore.

Se l'M20 entra in Stato Editor, l'utente può incominciare a modificare la linea (cancellando, inserendo e rimpiazzando i caratteri) facendo uso dei comandi dello Stato Editor (vedere Capitolo 3).

In Stato Editor il BASIC prende in esame i caratteri introdotti da tastiera di volta in volta che vengono impostati, senza aspettare che l'utente prema **CR**.

Impostando **CR** il BASIC esce dallo Stato Editor.

Errori di Sintassi

Se durante l'esecuzione di una linea di programma viene riscontrato un errore sintattico, il sistema M20 visualizza:

Syntax error in nn...n

Ok

nn...n

e automaticamente entra in Stato Editor.

Qui nn...n sta a indicare il numero di linea dove è stato riscontrato l'errore.

Stati dell'Editor

L'Editor include i seguenti Stati:

- Stato Cancellazione
- Stato Modifica
- Stato Inserimento

Per entrare e uscire da questi Stati l'utente deve usare i corrispondenti comandi dell'Editor (vedere Capitolo 3).

ISTRUZIONI E COMANDI BASIC

Risulta a volte difficile distinguere un'istruzione BASIC da un comando BASIC, poichè entrambi possono essere usati in un programma o in una linea immediata, ma:

- le istruzioni BASIC sono generalmente usate in linee di programma e vengono impostate in sequenza per formare un programma (eccezione fatta per PRINT, LPRINT, LET e SWAP che sono anche usate spesso in linee

immediate, quando viene usato l'M20 come una macchina calcolatrice sia per visualizzare subito il risultato di un calcolo, sia per verificare se un programma è corretto visualizzando il contenuto delle variabili di programma)

- I comandi BASIC sono usati per la gestione dei files o per lanciare programmi di servizio (utilities) come per esempio quando si listano i programmi o si azzera la memoria. Essi sono generalmente usati in linee immediate eccezione fatta per KILL, LOAD, RUN, SYSTEM, TROFF, TRON e WIDTH che sono usati con una certa frequenza anche in un programma.

Nel seguito del manuale, quando verrà introdotta una istruzione o un comando o una funzione BASIC, scriveremo a fianco del suo nome:

- (IMMEDIATO), se può essere usato solo in modo immediato
- (PROGRAMMA), se può essere usato solo in un programma
- (PROGRAMMA/IMMEDIATO), se può essere usato sia in un programma, sia in modo immediato.

COME CAMBIARE MODALITA' O AMBIENTE

E' possibile cambiare modalità operative (Stato Comandi, Esecuzione, Editor) o ambiente (PCOS o BASIC) se l'utente imposta un apposito comando o se si verificano determinate condizioni.

La seguente tabella indica tutti i casi possibili.

SE l'M20 è in...	E SE...	ALLORA...
BASIC (Stato Esecuzione)	l'utente imposta CTRL C mentre l'M20 sta eseguendo un programma BASIC o una linea a esecuzione immediata	l'esecuzione viene interrotta e l'M20 entra nello Stato Comandi BASIC

	l'utente imposta: CTRL RESET	la memoria viene azzerata e il PCOS viene nuovamente caricato in memoria
	viene riscontrato un errore sintattico	l'M20 entra in Stato Editor alla linea che ha causato l'errore
	l'esecuzione di un programma o di un comando BASIC viene portata a termine OPPURE viene riscontrato un errore (escluso un errore sintattico) OPPURE viene incontrata una istruzione STOP (o END)	l'M20 entra in Stato Comandi BASIC
BASIC (Stato Comandi)	l'utente imposta una linea immediata	l'M20 entra in Stato Esecuzione (sempre in ambiente BASIC), esegue la linea immediata e ritorna in Stato Comandi BASIC.
	l'utente imposta S Y S T E M CR Nota: SYSTEM può anche essere usato in un programma BASIC.	l'M20 entra in PCOS e viene azzerata sia la memoria a disposizione dell'utente, sia la memoria dove è allocato il BASIC.

	l'utente imposta E D I T SPACE n n ... n CR oppure: E D I T SPACE . CR	l'M20 entra in Stato Editor di Linea
	l'utente preme: CTRL RESET	la memoria viene azzerata e il PCOS viene nuovamente caricato in memoria
Stato Editor di Line	l'utente preme CR	l'M20 entra in Stato Comandi BASIC.(L'ultima linea modificata viene visualizzata). Le variabili di programma vengono azzerate (cioé le variabili numeriche poste a zero e le variabili stringa poste uguali alla stringa nulla).
	l'utente preme E (Exit)	l'M20 entra in Stato Comandi BASIC. (La parte restante dell'ultima linea modificata non viene visualizzata). Le variabili di programma vengono azzerate.
	l'utente preme Q (Quit Editing)	l'M20 entra in Stato Comandi BASIC e cancella tutte le eventuali modifiche fatte sulla linea. Le variabili di programma mantengono il loro valore.

	l'utente preme CTRL RESET	la memoria viene azzerata e il PCOS viene nuovamente caricato in memoria
PCOS	l'utente imposta B A CR	il BASIC viene caricato in memoria e l'M20 entra in Stato Comandi BASIC
	l'utente imposta qualsiasi altro comando PCOS	l'M20 entra in Stato Esecuzione (sempre in ambiente PCOS).
	l'utente imposta: CTRL RESET	la memoria viene azzerata e il PCOS viene nuovamente caricato in memoria
	l'utente imposta un identificatore di file (corrispondente a un programma BASIC) che sia stato registrato con l'estensione BAS (per es. FILEA.BAS) OPPURE l'utente imposta il comando BASIC seguito da un identificatore di file, corrispondente ad un programma BASIC (per es. BA FILEB)	L'M20 entra in BASIC (Stato Esecuzione) e viene eseguito il programma specificato

2. INTRODUZIONE, LISTING ED ESECUZIONE DI UN PROGRAMMA

SOMMARIO

Questo capitolo illustra la sintassi adottata nel manuale per descrivere la struttura di una istruzione o di un comando o di una funzione; viene quindi spiegato come documentare un programma, come introdurlo in memoria da tastiera, come listarlo, trasferirlo da memoria a disco e viceversa e come eseguirlo.

INDICE

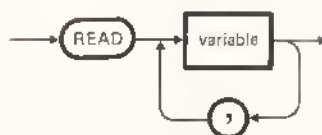
SINTASSI ADOTTATA

La sintassi adottata per descrivere la struttura di un comando, di un'istruzione o di una funzione BASIC si basa sui diagrammi sintattici (syntax diagrams).

Un diagramma sintattico è un grafo con una sola entrata ed una sola uscita. Ogni percorso attraverso il grafo rappresenta una possibile sequenza di simboli.

La seguente tabella enumera le regole da seguire per disegnare un diagramma sintattico.

N°	REGOLA	ESEMPIO
1	tutte le parole e tutti i simboli racchiusi in ovali o in cerchi devono essere impostati esattamente come indicato nella sintassi; tutte le parole racchiuse in un rettangolo rappresentano nomi di parametri usati in istruzioni o comandi o funzioni; il significato di ogni parametro viene descritto nel testo che segue alla sintassi (quando non è di immediata interpretazione).	RUN, line number, file identifier
2	una diramazione indica una scelta: l'utente deve scegliere uno dei percorsi. Per esempio, dopo RUN è possibile: - impostare un numero di linea OPPURE	RUN, line number, file identifier

	- un identificatore di file	
3	una diramazione senza parametri indica che l'alternativa è un "bypass" (vedere nell'esempio il percorso che supera l'opzione ,R)	... , R ...
4	un ciclo di ritorno (loop) indica che un parametro può essere ripetuto. Per esempio, il parametro "variable" può essere ripetuto n volte in un'istruzione READ, e ogni "variable" deve essere separata dalla successiva da una virgola	READ variable ,
5	questo manuale indica in lettere maiuscole le parole riservate del BASIC, anche se l'utente può impostarle in lettere minuscole. Alcuni esempi di parole riservate sono indicati a lato (sono le parole chiave del nostro programma RETTANGOLO1)	REM INPUT IF...THEN LET PRINT GOTO END . . .
6	anche i nomi dei comandi PCOS sono mnemonici. Per esempio: BASIC - va in BASIC VCOPY - copia un volume ecc. L'utente può introdurli in lettere minuscole o in maiuscole. Vengono normalmente introdotti in lettere minuscole e in forma abbreviata (solo le prime due lettere), come indicato nel programma sintattico qui a lato	ba file identifier

COME DOCUMENTARE UN PROGRAMMA

Spesso l'utente desidera inserire commenti in un programma per renderlo più leggibile e di più immediata comprensione. Ciò può essere fatto in due modi:

- utilizzando istruzioni REM

- utilizzando campi commenti.

REM/CAMPI COMMENTI (PROGRAMMA)

L'istruzione REM (Remark) è una delle possibilità offerte dal BASIC per documentare un programma. L'utente può inserire qualsiasi stringa di caratteri dopo REM. Un'altra possibilità offerta dal BASIC è quella di inserire un campo commento, cioè una stringa di caratteri preceduta da un apostrofo (') e chiusa da **CR**.

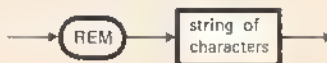



Figura 2-1 Istruzione REM

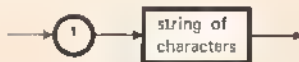



Figura 2-2 Campo Commento

Esempi

SE l'utente imposta. . .	ALLORA...
1Ø REM RETTANGOLO1 **CR**	una istruzione REM dà il titolo al vostro programma. E' buona norma di programmazione dare sempre un titolo a ogni programma
1ØØ REM SUBROUTINE1 **CR**	una istruzione REM segna l'inizio di una subroutine (vedere capitolo 1Ø). E' buona norma di programmazione dare sempre un titolo a ogni subroutine
1Ø 'RETTANGOLO1 **CR**	un campo commento dà il titolo al vostro programma. Nota: In questo caso l'apostrofo ha la stessa funzione della parola REM, dato che il campo commento occupa un'intera linea
15Ø LET A=(A1+A2)/2 'Media **CR**	un campo commento è inserito a fine istruzione

Note

Una istruzione REM non può essere seguita da altre istruzioni su una stessa linea, ma può essere l'ultima istruzione di una linea che contiene più istruzioni.

Un campo commento può:

- occupare un'intera linea (in tale caso l'apostrofo ha la stessa funzione di una istruzione REM)

- essere inserito a fine istruzione.

Sia le istruzioni REM, sia i campi commento possono essere inseriti in qualsiasi punto del programma. Non sono istruzioni eseguibili, ma appaiono nel listing.

INTRODUZIONE DI UN PROGRAMMA DA TASTIERA

Il messaggio OK é sul video. Il BASIC é in attesa che l'utente introduca dei caratteri da tastiera. E' possibile iniziare subito introducendo la prima istruzione iniziando a impostare il numero di linea (ad esempio 1Ø). Però l'utente può anche richiedere che il sistema faccia la numerazione automatica di linee. Questa richiesta viene fatta tramite il comando AUTO (descritto in seguito).

Però se l'utente ha già introdotto un programma e vuole introdurne un altro, deve, per prima cosa, impostare il comando NEW, che azzera la memoria.

Il programma può anche essere cancellato dalla memoria sia caricando un nuovo programma da disco, sia introducendo un comando SYSTEM (per tornare al PCOS), sia spegnendo la macchina. In questi casi, é opportuno che l'utente, in precedenza, registri il programma su disco (a meno che già non ne abbia una copia).

Impostiamo il programma (RETTANGOLO1) in tastiera. Per prima cosa impostiamo:

N E W SPACE CR

Poi introduciamo:

```
1Ø REM RETTANGOLO1   CR
2Ø INPUT "Base";B CR
3Ø IF B<=Ø THEN 2Ø CR
4Ø INPUT "Altezza";H CR
5Ø IF H<=Ø THEN 4Ø CR
6Ø LET AREA = B*H CR
7Ø PRINT "Area=";AREA;"B=";B;"H=";H CR
8Ø GOTO 20 CR
9Ø END CR
```

Conviene usare un intervallo di 1Ø fra ogni numero di linea. Questo permette all'utente di modificare facilmente un programma inserendo istruzioni fra le linee già esistenti.

Le linee di programma sono ordinate in memoria secondo la sequenza del numero di linea, non nell'ordine in cui sono state introdotte in tastiera.

Per esempio, se impostiamo prima l'istruzione 5Ø, poi la 1Ø ecc... fino a impostarle tutte ma in un ordine qualsiasi noi introduciamo in memoria il medesimo programma.

L'utente può introdurre parole chiave e nomi di variabili sia con lettere maiuscole che con lettere minuscole. Queste ultime verranno convertite nelle corrispondenti maiuscole quando si richiede il listing del programma.

AUTO (IMMEDIATO)

Fa iniziare la numerazione automatica delle linee.

Figura 2-3 Comando AUTO

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO	VALORI DI DEFAULT
line number	è il primo numero di linea generato	1Ø se viene omesso anche il parametro interval; in ogni altro caso Ø.
—	il primo numero di linea generato è il numero della linea attuale	1Ø
interval	è l'intervallo tra i numeri di linea	1Ø

Esempi

SE l'utente imposta...	ALLORA la numerazione di linea inizia...	E l'intervallo di linea è...
AUTO CR	alla linea 1Ø (valore di default)	1Ø (valore di default)
AUTO .,3Ø CR	alla linea attuale	3Ø
AUTO 1ØØ CR	alla linea 1ØØ	1Ø (valore di default)
AUTO 15Ø, CR	alla linea 15Ø	l'ultimo intervallo specificato da un precedente comando AUTO oppure 1Ø il valore di default) se non é stato impostato un comando AUTO in precedenza.
AUTO 2ØØ,2Ø CR	alla linea 2ØØ	2Ø
AUTO , 3 CR	alla linea Ø	3

Un Asterisco dopo un Numero di Linea

SE...	ALLORA...
AUTO genera un numero di linea già esistente	un asterisco compare sul video dopo il numero di linea per avvertire l'utente che egli sta per sostituire una linea già esistente. Comunque, premendo CR immediatamente dopo l'asterisco, si lascia inalterata la linea esistente e si genera il prossimo numero di linea. Nota: Questo può succedere solo se si introduce AUTO quando un programma è già esistente in memoria.

Per Terminare la Numerazione Automatica

SE l'utente imposta...	ALLORA...
CTRL C	la numerazione automatica viene interrotta e l'M20 entra in Stato Comandi. Nota: Quando si imposta CTRL C la linea attuale viene cancellata

NEW (PROGRAMMA/IMMEDIATO)

Cancella il programma residente in memoria e le variabili, permettendo all'utente di introdurre un nuovo programma.

NEW setta a zero il bit di "trace" (tracciamento) allo stesso modo del comando TROFF (vedere capitolo 13) e chiude tutti i file dati (vedere capitolo 12).

Figura 2-4 Comando NEW

Esempi

SE l'utente imposta...	ALLORA...
NEW CR	il programma residente in memoria viene cancellato, insieme alle sue variabili.
1Ø REM RETTANGOLO1 CR 2Ø INPUT "Base";B CR . . .	L'utente introduce un nuovo programma da tastiera.

Nota: Non è necessario impostare NEW prima di trasferire in memoria un programma da disco, tramite il comando LOAD o RUN (questi comandi azzerano la memoria automaticamente).

COME LISTARE UN PROGRAMMA

Quando un programma è in memoria, può essere listato. Per listare un programma, l'utente può introdurre il comando LIST (il "listing" comparirà sul video), oppure il comando LLIST (il "listing" comparirà su stampante). L'utente non può listare un programma protetto (registrato con l'opzione P).

Quando si lista un programma le parole riservate (parole chiave, nomi di variabile e nomi di funzione) che l'utente aveva impostato in lettere minuscole, vengono convertite in lettere maiuscole e il punto interrogativo (?) eventualmente usato al posto della parola chiave PRINT viene convertito in PRINT. Inoltre le linee vengono visualizzate per numero di linea crescente indipendentemente dall'ordine con cui sono state impostate.

Per listare il programma RETTANGOLO1 sul video, impostiamo **L** **I** **S** **T** **CR**. Sul video apparirà:

```
LIST
10 REM RETTANGOLO1
20 INPUT "Base";B
30 IF L<=0 THEN 20
40 INPUT "Altezza";H
50 IF H<=0 THEN 40
60 LET AREA=B*H
70 PRINT "Area=";AREA;" B=";B;" H=";H
80 GOTO 20
90 END
OK
```

Alla fine della lista l'M20 entra in Stato Comandi e visualizza OK.

LIST/LLIST (IMMEDIATI)

LIST lista le linee di programma sul video, LLIST lista le linee di programma su stampante.

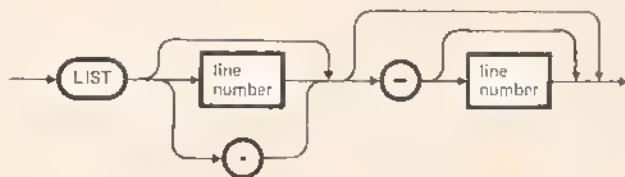



Figura 2-5 Comando LIST

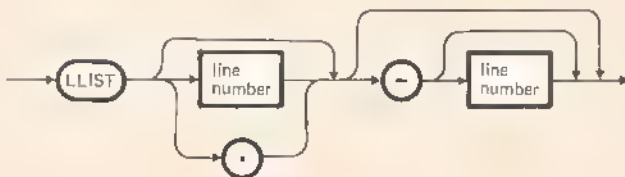



Figura 2-6 Comando LLIST

Esempi

SE l'utente imposta...	ALLORA...
LIST CR	l'intero programma viene listato
LIST 15Ø CR	la linea 15Ø viene listata
LIST . CR	la linea attuale viene listata
LIST 2ØØ- CR	vengono listate la linea 2ØØ e tutte le successive
LIST -1ØØØ CR	vengono listate tutte le linee dall'inizio alla linea 1ØØØ
LIST 1ØØ-19Ø CR	vengono listate tutte le linee dalla linea 1ØØ alla 19Ø
LIST .-5ØØ CR	vengono listate tutte le linee dalla linea attuale alla 5ØØ

Per Interrompere una Lista

SE...	ALLORA...
l'utente imposta CTRL S	la lista del programma viene sospesa, senza che il sistema entri in Stato Comandi. L'utente può continuare la lista premendo un qualsiasi tasto
l'utente imposta: CTRL C	l'M20 entra in Stato Comandi interrompendo definitivamente la lista
l'intero programma è stato listato	l'M20 entra in Stato Comandi

FILE PROGRAMMA E FILE DATI

Un file è una sequenza di istruzioni (file programma) o di dati (file dati), sequenza che può essere memorizzata su disco.

La seguente tabella sintetizza le caratteristiche principali di un file programma e di un file dati.

TIPO DI FILE	SIGNIFICATO
file programma	un file programma è una sequenza di linee di programma. Queste vengono memorizzate secondo la successione crescente dei numeri di linea, indipendentemente dall'ordine in cui sono state impostate. Un file programma viene memorizzato in un formato binario "compatto" e registrato su disco o in questo formato o in un formato "sorgente" ASCII (se l'utente usa l'opzione A per registrarlo). I file in formato ASCII sono sequenze di caratteri ASCII e rappresentano effettivamente la sequenza di caratteri che appare su video quando l'utente lista il programma. Quando un programma viene trasferito da disco a memoria (tramite un comando LOAD o RUN), esso viene sempre convertito in formato "compatto".
file dati	un file dati è una sequenza di dati numerici e/o stringa registrata su disco. Un file dati viene creato tramite un programma BASIC. Per prima cosa deve essere aperto tramite una istruzione OPEN che ne specifichi il nome, il metodo d'accesso e associ al file un numero da 1 a 15. Ogni successiva istruzione di Input/Output nel programma farà riferimento al file tramite questo numero. Quando il programma non deve più leggere o registrare dati sul file è buona norma di programmazione "chiuderlo" tramite una istruzione CLOSE. In ogni caso tutti i file dati vengono chiusi quando il programma termina con una istruzione END.

> Nota: Chiudere un file dati significa impedirne l'accesso da programma. Il programma, per poter accedere di nuovo a quel file deve riaprirlo tramite un'altra istruzione OPEN associandovi un nuovo (o lo stesso) metodo d'accesso e un nuovo (o lo stesso) numero di file. Soltanto il nome dovrà essere lo stesso.

IDENTIFICATORI DI FILE E DI VOLUME

Un disco può contenere uno o più file programmi e/o file dati. Un singolo file non può essere registrato parte su un disco e parte su un altro.

L'insieme di file registrati su un unico disco è denominato "volume".

Ogni file e ogni volume hanno un proprio identificatore. Non è possibile avere due file con lo stesso nome su uno stesso volume. Se si registra un file programma su un disco dove esiste già un file con lo stesso nome, il nuovo file sostituisce il precedente.

Si può assegnare un identificatore ad un file sia tramite l'istruzione OPEN (file dati), sia tramite il comando SAVE (file programma) e sia tramite il comando FNEW del PCOS.

Si può assegnare un identificatore a un volume o con il comando VFORMAT o con il comando VNEW o con il comando VRENAME del PCOS.

Il sistema riconosce un identificatore di volume e può accedere a uno dei suoi file soltanto se il dischetto corrispondente è stato inserito in una delle due unità disco. Questa limitazione non vale per l'hard disk, dato che questa unità è sempre in linea.

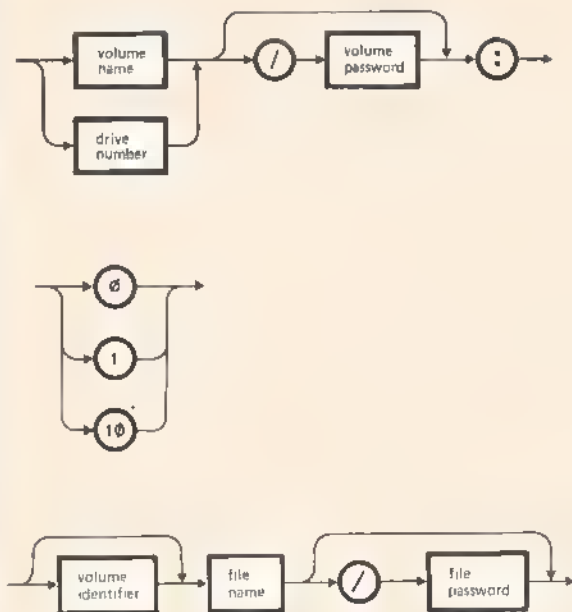



Figura 2-7 Identificatore di File e di Volume

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
volume name (nome di volume)	stringa di (al massimo 14) caratteri stampabili ASCII (alcuni caratteri non sono ammessi, vedere Caratteri Illegali in questo paragrafo). Per selezionare un volume in un comando PCOS o BASIC o in una istruzione OPEN, l'utente deve specificare un nome di volume oppure un numero di unità disco.

Il nome di volume (come pure il numero di unità disco) può essere seguito da una password di volume. Alla fine di un identificatore di volume, l'utente deve impostare il carattere due punti (:).

Per esempio:

SAVE "VOL1:FILE1" **CR**

Qui VOL1 è un nome di volume; FILE1 è un nome di file e l'intera stringa VOL1:FILE1 è un identificatore di file. Tramite questo comando l'utente registra il file programma FILE1 sul disco che ha nome VOL1. (Per ulteriori dettagli vedere il comando SAVE in questo capitolo).

Nota: Quando l'utente specifica un identificatore di file o di volume in una istruzione o in un comando BASIC, deve racchiudere l'identificatore tra virgolette ("), oppure scrivere una variabile stringa, o una espressione stringa il cui valore è l'identificatore.

Quando specifica un identificatore di file o di volume in un comando PCOS, non deve racchiudere l'identificatore tra virgolette.

Per esempio:

vn VOL1: **CR**

Si noti inoltre che tutti i comandi e le istruzioni BASIC possono solo specificare un identificatore di volume come facente parte di un identificatore di file. Fa eccezione il solo comando FILES, quando viene utilizzato per visualizzare l'elenco di tutti i file di un volume.

Per esempio:

FILES "VOL2:" **CR**

drive number (numero unità disco)

il drive number può essere Ø (che indica l'unità, alla destra), oppure 1 (che indica l'unità alla sinistra), oppure 1Ø (che indica l'hard disk). In un sistema hard disk il drive Ø è a sinistra e il drive 1 non esiste.

Per esempio:

LOAD "1:FILEØØ2" **CR**

Qui 1: indica che il file FILEØØ2 risiede sul dischetto inserito nella unità 1. Il comando trasferisce il programma da disco a memoria (per ulteriori dettagli vedere il comando LOAD in questo capitolo)

file password

OPPURE

volume password

stringa di (al massimo) 14 caratteri stampabili ASCII. Alcuni caratteri sono esclusi (vedere Caratteri Illegali in questo paragrafo).

Una password permette di proteggere un volume o un file limitandone l'accesso a chi conosca la password. Una password può essere impostata dopo un nome di volume, o un numero di unità disco, o un nome di file e deve essere preceduta da una barra (/).

Per esempio:

RUN "Ø:RETTANGOLO1/R1" **CR**

Con questo comando l'utente trasferisce da disco a memoria il programma RETTANGOLO1 che ha la password R1 e ne lancia l'esecuzione . RETTANGOLO1 è registrato sul dischetto inserito nell'unità Ø. (Per ulteriori dettagli vedere il comando RUN in questo capitolo)

file name (nome di file)

stringa di (al massimo) 14 caratteri stampabili ASCII. Alcuni caratteri sono esclusi (vedere Caratteri Illegali in questo paragrafo).

Per selezionare un file in un comando PCOS o BASIC o in una istruzione OPEN l'utente deve specificare un nome di file. Esso può essere preceduto da un identificatore di volume e

seguito da una estensione (extension) e/o da una password.

Per esempio:

SAVE "1:PRIMENUMBERS/PN" **CR**

Tramite questo comando l'utente registra il file programma PRIMENUMBERS sul dischetto inserito nella unità 1 e dà al file la password PN.

Nota: Se l'utente non specifica un identificatore di volume prima del nome di file, il sistema seleziona l'unità selezionata per ultima oppure l'unità Ø se nessuna unità era stata, in precedenza, selezionata per ultima.

Nota: Il nome di file può includere un'estensione, cioè una stringa di (al massimo) 12 caratteri stampabili ASCII, preceduta da un punto decimale (.). Alcuni caratteri sono esclusi (vedere Caratteri Illegali in questo paragrafo).

Nota: filename.extension non può superare 14 caratteri in totale (compreso il punto)

Per esempio:

LOAD "FILEA.CHAR" **CR**

trasferisce il programma FILEA che ha l'estensione CHAR da disco in memoria. Esso risiede sull'ultima unità selezionata.

Nota: Alcune estensioni hanno un significato particolare: BAS (programmi BASIC); CMD (Comandi PCOS transienti); SAV (Comandi PCOS transienti, che divengono residenti la prima volta che vengono eseguiti). Per ulteriori dettagli vedere il "Professional Computer Operating System (PCOS) Guida Utente".

Caratteri Illegali

asterisco	(*)	barra rovescia	(\)	due punti	(:)
più	(+)	segno di numero	(#)	punto interrogativo	(?)
trattino	(-)	uguale	(=)	virgolette	(")
barra	(/)	virgola	(,)	apice	(')
spazio					

oppure un qualsiasi carattere di controllo

PASSWORD

L'utente può proteggere un volume o un file tramite una password (parola d'ordine).

Una password può essere usata per abilitare un volume: un volume è abilitato se non ha password oppure se l'utente ha specificato questa password in un comando BASIC o PCOS.

L'utente deve specificare la password in modo corretto tutte le volte che nomina un volume o un file ai quali è stata assegnata una password.

Nota: Né il BASIC né il PCOS consentono di venire a conoscere una password nell'eventualità che l'utente l'abbia dimenticata.

PASSWORD DI VOLUME

SE l'utenta vuole...	ALLORA...
assegnare una password a un voluma	dave impostare il comando VPASS del PCOS specificando la password. Per esempio: vp MYVOL:,MYPASS CR SE il volume ha già una password, questa dave essar specificata con il comando VPASS, che, in questo caso, cambierà la password. Par asampio: vp VOL1/OLDPASS:,NEWPASS CR
accedera a un voluma che ha una password (o a un suo fila)	dava abilitare quel volume, specificandone la password dopo il nome di volume o dopo il numero dell'unità disco, in un comando BASIC o PCOS oppure in una istruzione OPEN. Nota: Una volta che una password di un dischetto è stata specificata, non è più necessario specificarla un'altra volta finché il dischetto non venga rimosso a un'altro dischatto vanga selezionato sulla stessa unità. Per l'hard disk, una volta cha la password à stata specificata, non à più nacassario specificarla un'altra volta finchè il PCOS non venga reinizializzato.
rimuovere una password da un volume	deve impostare il comando VDEPASS del PCOS Nota: L'utente deve conoscere la password par poter usare il comando VDEPASS.
impedira la visualizzazione di una password di volume	deve impostare CTRL G. Il cursore cambia forma e fraquanza di lampeggio a i caratteri impostati successivamente non vengono visualizzati (Hide State). Par ritornare a visualizzare i carattari impostati (Display State), l'utente deve impostare di nuovo CTRL G oppure CR

SE l'utente vuole	ALLORA ....
assegnare una password a un file	deve impostare il comando FPASS del PCOS specificando la password. SE il file ha già una password, questa deve essere specificata con il comando FPASS che, in questo caso, cambierà la password.
assegnare una password a un file programma (che ne é sprovvisto)	può impostare il comando FPASS del PCOS oppure il comando SAVE del BASIC specificandone la password: Per esempio: SAVE "FILEABC/PASSABC CR"
accedere a un file che ha una password	deve specificare la password dopo il nome del file. Per esempio: LOAD "FILEZ/PASSZ1" CR Se anche il volume ha una password, l'utente deve specificarla (a meno che il volume sia già stato abilitato)
rimuovere una password da un file	deve impostare il comando FDEPASS del PCOS Nota: L'utente deve conoscere la password del file per poterla rimuovere o cambiare
impedire la visualizzazione di una password di file	deve impostare CTRL G. Il cursore cambia forma e frequenza di lampeggio e i caratteri impostati non vengono visualizzati (Hide State). Per ritornare a visualizzare i caratteri impostati, l'utente deve premere di nuovo CTRL G oppure CR.

PROTEZIONE DA SCRITTURA

L'utente può proteggere da scrittura sia un intero dischetto che un singolo file.

SE l'utente vuole ...	ALLORA ...
proteggere da scrittura un volume (cioé evitare qualsiasi operazione di scrittura su quel dischetto)	deve proteggere l'apposito intaglio con una etichetta metallizzata. Nota: Non è possibile proteggere da scrittura l'hard disk, ma solo i file residenti su di esso.
togliere la protezione da scrittura a un dischetto	oeve togliere l'etichetta metallizzata
proteggere da scrittura un file	deve impostare il comando FWPROT del PCOS, specificando l'identificatore di file.
togliere la protezione da scrittura a un file	deve impostare il comando FUNPROT del PCOS, specificando l'identificatore di file

REGISTRAZIONE DI UN PROGRAMMA

Un programma resta in memoria finché la macchina é accesa. Appena si spegne la macchina, la memoria viene azzerata e si perde quindi il programma. Perciò se l'utente vuole conservare una copia del programma che ha appena introdotto in memoria da tastiera, deve impostare il comando SAVE (che registra il programma su disco).

Può essere utile registrare il programma anche in altre occasioni che vengono indicate nella seguente tabella. In ognuno dei seguenti casi si suppone che il disco sia abilitato, altrimenti l'utente deve specificare la password di volume mediante il comando SAVE. Inoltre se si vuole registrare il programma su un dischetto, questo non deve ovviamente essere protetto da scrittura.

SE l'utente vuole...	ALLORA...
spegnere la macchina	deve registrare il programma residente in memoria (supposto che non ne esista già una copia su disco)
introdurre un altro programma da tastiera	deve registrare il programma residente in memoria (supposto che non ne esista già una copia su disco)
trasferire un altro programma da disco a memoria (tramite un comando LOAD o RUN)	deve registrare il programma residente in memoria (supposto che non ne esista già una copia su disco).
passare in PCOS (tramite il comando SYSTEM)	deve registrare il programma residente in memoria (supposto che non ne esista già una copia su disco).
sostituire la versione precedente del suo programma	deve registrare il programma residente in memoria, specificando lo stesso nome della versione precedente E la stessa password (se la versione precedente aveva una password)
registrare in formato ASCII il programma residente in memoria	deve specificare l'opzione A nel comando SAVE
proteggere il programma residente in memoria contro ogni tentativo di listarlo, modificarlo, o registrarlo di nuovo	deve specificare l'opzione P nel comando SAVE

Nota: Durante un'operazione di registrazione la luce rossa dell'unità disco si accende. Quando si spegne, il programma è registrato e il messaggio Ok appare su video.

SAVE (PROGRAMMA/IMMEDIATO)

Registra il programma residente in memoria su disco e gli assegna un nome ed eventualmente una password

Figura 2-8 Comando SAVE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file identifier	può essere sia una costante che una variabile stringa. Specifica il nome del programma da registrare su disco. Può includere una password di file e un identificatore di volume
A	specifica che il programma deve essere registrato su disco in formato ASCII
P	specifica che il programma deve essere registrato su disco protetto contro ogni tentativo di listarlo, modificarlo o registrarlo di nuovo

Esempi

In ognuno dei seguenti esempi si suppone che il disco sia abilitato e non sia protetto da scrittura.

SE l'utente imposta...	ALLORA...
SAVE "RETTANGOLO1" CR	il programma RETTANGOLO1 viene registrato sul disco inserito nell'ultima unità selezionata. RETTANGOLO1 non ha password
SAVE "Ø:RETTANGOLO1" CR	il programma RETTANGOLO1 viene registrato sul dischetto inserito nella unità Ø. RETTANGOLO1 non ha password
SAVE "1Ø:RETTANGOLO1" CR	il programma RETTANGOLO1 viene registrato sull'hard disk. RETTANGOLO1 non ha password
SAVE "VOL1:RETTANGOLO1" CR	il programma RETTANGOLO1 viene registrato su VOL1, che può essere inserito in entrambe le unità (dato che l'utente ha specificato il nome del volume). RETTANGOLO1 non ha password
SAVE "VOL1:R1/PASS" CR	R1 viene registrato su VOL1, che può essere inserito in entrambe le unità e assegna la password PASS al file R1.

Sostituzione di un file

Si suppone, negli esempi che seguono, che il volume sia abilitato e non sia protetto da scritture.

SE l'utente imposta...	E SE...	ALLORA...
SAVE "FILE1"	FILE1 esiste già sul disco selezionato E non ha password	il programma residente in memoria sostituirà la versione precedente con lo stesso nome.

SAVE "FILE1/PASS1"	FILE1 esiste già sul disco selezionato E ha password PASS1	il programma residente in memoria sostituirà la versione precedente con lo stesso nome e la stessa password
	FILE1 esiste già sul disco selezionato E ha una password differente	non avviene sostituzione, e il sistema emette un messaggio d'errore (vedere Appendice C)
	FILE1 esiste già sul disco selezionato E non ha password	il programma residente in memoria sostituirà la versione precedente con lo stesso nome, e l'attuale versione avrà la password PASS1

Opzione A

Se l'utente specifica l'opzione A, il file viene registrato in formato ASCII.

Se egli, invece, non specifica l'opzione A (cioè nessuna opzione o l'opzione P), il file viene registrato in formato binario "compatto".

Il formato ASCII occupa più spazio su disco che il formato "compatto", ma alcuni comandi richiedono che i file siano in formato ASCII, come ad esempio il comando MERGE.

SE l'utente imposta...	ALLORA...
SAVE "GEOMETRY",A CR	il programma GEOMETRY viene registrato in formato ASCII (cioè in una sequenza di caratteri ASCII) sul disco inserito nell'ultima unità selezionata GEOMETRY non ha password. Si suppone che il disco sia abilitato.

Opzione P

Se l'utente specifica l'opzione P, il file non è soltanto registrato in formato binario "compatto", ma anche "protetto" contro ogni tentativo di:

- listarlo

- modificarlo

- registrarlo di nuovo

Nota: L'utente non può rimuovere la protezione P.

SE l'utente imposta...	ALLORA...
SAVE "Ø:GEODESY",P CR	GEODESY viene registrato in formato binario "compatto" e "protetto" sul disco inserito nella unità Ø. GEODESY non ha password. Si suppone che il disco sia abilitato.

TRASFERIMENTO DA DISCO A MEMORIA

Se il programma che l'utente vuole portare in memoria risiede su disco, l'utente deve impostare un comando LOAD.

LOAD, nel trasferire un programma in memoria, ricopre qualsiasi altro programma ivi residente, perciò, prima di impostare un comando LOAD, l'utente dovrà provvedere a registrare su disco il programma attuale (supposto che non ne abbia già una copia).

Per accedere a un programma su disco tramite un comando LOAD, il disco deve essere abilitato (altrimenti l'utente deve specificare la password). Per accedere ad un programma dotato di password, l'utente deve specificarne la password sempre tramite il comando LOAD.

Se l'utente specifica l'opzione R tutti i file dati che erano stati aperti dal programma ricoperto rimangono aperti, inoltre il nuovo programma non solo viene trasferito in memoria ma viene anche eseguito (senza che l'utente debba impostare RUN).

LOAD (PROGRAMMA/IMMEDIATO)

Trasferisce un programma da disco in memoria e ne lancia l'esecuzione (se l'utente specifica l'opzione R).

Figura 2-9 Comando LOAD

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file identifier	può essere sia una costante che una variabile stringa; specifica il file programma, che deve essere trasferito in memoria da disco
R	specifica che tutti i file dati aperti dal programma precedente vengano lasciati aperti e che il nuovo programma, dopo essere stato trasferito da disco in memoria, venga anche eseguito

Esempi

SE l'utente imposta...	ALLORA...
LOAD "1Ø:RETTANGOLO1" CR	il programma RETTANGOLO1 viene trasferito in memoria dall'hard disk. Si suppone che RETTANGOLO1 non abbia password e che il disco sia abilitato

LOAD "VOL1:RETTANGOLO1/P1" CR	il programma RETTANGOLO1 viene trasferito da disco a memoria. RETTANGOLO1 ha come password P1 e risiede sul disco VOL1 che può essere inserito sia nella prima che nella seconda unità. Si suppone che VOL1 sia abilitato.
LOAD "V3/P3:FAA" CR	il programma FAA viene trasferito da disco a memoria. FAA risiede sul disco V3 che ha la password P3. Il disco V3 può essere inserito sia nella prima che nella seconda unità. Il comando LOAD, in questo caso, abilita anche il disco V3, dato che ne specifica la password. Si suppone che FAA non abbia password.
LOAD B$ CR	il programma, individuato dal contenuto della variabile B$, viene trasferito in memoria.

Opzione R

Se l'utente specifica l'opzione R, tutti i file dati che erano stati aperti dal programma precedentemente eseguito restano aperti e viene anche lanciata l'esecuzione del nuovo programma trasferito da disco in memoria (senza che l'utente debba impostare il comando RUN).

Se l'utente non specifica l'opzione R, il comando LOAD chiude tutti i file dati che erano stati aperti dal programma precedentemente eseguito.

Si noti che:

LOAD file identifier, R CR

e RUN file identifier, R CR

sono comandi equivalenti

SE l'utente imposta...	ALLORA...
LOAD "ACCOUNT",R CR	il programma ACCOUNT viene trasferito in memoria da disco e viene eseguito; tutti i file che erano stati aperti dal programma precedente restano aperti. ACCOUNT risiede sull'unità selezionata per ultima. ACCOUNT non ha password e il disco su cui risiede deve essere abilitato.

ESECUZIONE DI UN PROGRAMMA

Una volta che un programma è in memoria può essere eseguito. Per dire al sistema M20 di eseguire un programma, si deve impostare un comando RUN, (oppure un comando LOAD con l'opzione R).

Il comando RUN esegue tutte le operazioni richieste dal programma residente in memoria, o trasferisce un programma da disco e poi le esegue. Quando il comando RUN specifica il nome di un file programma, può includere:

- una password di file, se il file ha una password

- una password di volume, se il volume ha una password (e non è stato ancora abilitato)

Se l'utente specifica l'opzione R, tutti i file dati che erano stati aperti dal programma precedente, restano aperti.

Prima di impostare RUN file-identifier (oppure RUN file-identifier,R), l'utente deve registrare su disco il programma residente in memoria (ammesso che non ne abbia già una copia).

Le istruzioni BASIC vengono eseguite seguendo la sequenza del numero di linea, a meno che un'istruzione di controllo (GOTO, ON...GOTO, IF... GOTO...ELSE, IF...THEN...ELSE, FOR/NEXT, WHILE/WEND) o il richiamo di un sottoprogramma (GOSUB, ON...GOSUB), specifichi altrimenti.

RUN (PROGRAMMA/IMMEDIATO)

Lancia l'esecuzione del programma residente in memoria. Se il comando RUN specifica il nome di un programma su disco, questo viene trasferito in memoria ed eseguito.

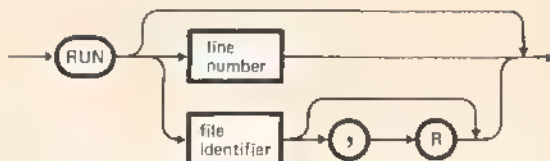



Figura 2-10 Comando RUN

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
line number	specifica il numero di linea da dove inizia l'esecuzione del programma. Se l'utente non specifica questo numero il programma viene eseguito a partire dalla prima istruzione Nota: RUN line number e GOTO line number hanno la stessa funzione salvo il fatto che RUN ha anche la funzione di azzerare le variabili (cioè porre a zero le variabili numeriche e assegnare la stringa nulla alle variabili stringa)
file identifier	può essere sia una costante che una variabile stringa; specifica il file programma che deve essere trasferito in memoria da disco ed eseguito
R	questa opzione specifica che tutti i file dati aperti dal programma precedente eseguito devono restare aperti. Se l'opzione non è specificata tutti i file dati vengono automaticamente chiusi prima di iniziare l'esecuzione del nuovo programma

Esempi

Negli esempi seguenti si suppone che il disco selezionato sia abilitato

SE l'utente imposta...	ALLORA...
RUN **CR**	il programma residente in memoria viene eseguito
RUN 15Ø **CR**	il programma residente in memoria viene eseguito a partire dalla istruzione 15Ø
RUN "1:Newfile" **CR**	il programma Newfile viene trasferito da disco in memoria ed eseguito. Il programma risiede sul dischetto inserito nella unità 1. Si suppone che Newfile non abbia password
RUN "NewVOL:Newfile" **CR**	il programma Newfile viene trasferito da disco a memoria ed eseguito. Newfile risiede sul disco NewVOL che può essere inserito nella prima o nella seconda unità. Si suppone che Newfile non abbia password
RUN "1:Newfile/NewPASS" **CR**	il programma Newfile viene trasferito da disco a memoria ed eseguito. Il programma ha la password NewPASS e risiede sul dischetto inserito nella unità 1.
RUN A$ **CR**	Il programma specificato tramite il contenuto della variabile A$ viene trasferito da disco in memoria ed eseguito

Opzione R

Se l'utente specifica l'opzione R nel comando RUN, tutti i file dati aperti dal programma eseguito in precedenza restano aperti.

Se l'utente non specifica l'opzione R, tutti i file dati vengono chiusi prima di lanciare l'esecuzione del nuovo programma.

Si noti che:

RUN file identifier, R CR

e

LOAD file identifier, R CR

sono comandi equivalenti

Se l'utente imposta...	ALLORA...
RUN "1Ø:Newfile",R CR	il programma Newfile viene trasferito da hard disk in memoria ed eseguito. I file dati aperti dal programma eseguito in precedenza restano aperti. Si suppone che Newfile non abbia password e che l'hard disk sia abilitato.

Interruzione dell'Esecuzione

SE ...	ALLORA ....
l'utente imposta CTRL C OPPURE viene eseguita una istruzione STOP	si ha una interruzione del programma, viene emesso il messaggio "Break in line nnnnn" e l'M20 entra in Stato Comandi. Nessun file dati viene chiuso. L'utente può visualizzare i valori delle variabili (tramite l'istruzione immediata PRINT) o modificare i valori (tramite l'istruzione immediata LET) L'utente può riprendere l'esecuzione impostando il comando CONT (a condizione che nessuna istruzione sia stata modificata).
viene riscontrato un errore (ad esclusione di un errore sintattico)	Si ha una interruzione del programma e l'M20 entra in Stato Comandi. Nessun file dati viene chiuso. L'utente può visualizzare i valori delle variabili, ma non può riprendere l'esecuzione.
viene riscontrato un errore sintattico	si ha una interruzione del programma, viene emesso il messaggio d'errore e l'M20 entra in Stato Editor di linea, alla linea che ha causato l'errore. L'utente può modificare la linea con i comandi dell'Editor, ma non può visualizzare le variabili (a meno che non entri in Stato Comandi premendo Q). L'esecuzione non può essere ripresa.

viene eseguita una istruzione END	si ha una interruzione del programma e l'M20 entra in Stato Comandi. Tutti i file dati vengono chiusi. L'utente può visualizzare i valori delle variabili (tramite l'istruzione immediata PRINT), ma non può riprendere l'esecuzione.

Arresto dell'Output su Video

SE ...	ALLORA ...
l'utente imposta CTRL S	L'output su video viene arrestato. L'utente può far riprendere l'output su video premendo un tasto qualsiasi. I file dati non vengono chiusi. L'utente non può visualizzare i valori delle variabili.

3. AGGIORNAMENTO E MODIFICA DI UN PROGRAMMA

SOMMARIO

Anche un buon programmatore ha spesso necessità di correggere e modificare i propri programmi. Un programma può essere modificato in vari modi: cancellando linee, sostituendo linee, inserendo linee, rinumerando le linee, modificando parti di linea con l'Editor.

Questo capitolo illustra tutte queste possibilità facendo uso del programma RETTANGOLO1. Verrà inoltre spiegato come rinominare un file, come cancellarlo su disco, come fare il MERGE di due programmi, come listare i nomi dei file di un disco. Si tenga presente che qualunque modifica a un programma, chiude tutti i file dati eventualmente ancora aperti, azzera le variabili numeriche e inizializza le variabili stringa col valore stringa nulla.

INDICE

COME CANCELLARE LE LINEE

A scopo dimostrativo useremo il programma RETTANGOLO1 già utilizzato nel capitolo 2.
Se il programma RETTANGOLO1 risiede in memoria, l'utente deve per prima cosa impostare il comando LIST.

VIDEO	COMMENTI
LIST 10 REM RETTANGOLO1 20 INPUT "Base";B 30 IF B<=0 THEN 20 40 INPUT "Altezza";H 50 IF H<=0 THEN 40 60 LET AREA=B*H 70 PRINT "Area=";AREA;" B=";B;" H=";H 80 GOTO 20 90 END Ok	RETTANGOLO1 ha due istruzioni di Input per introdurre da tastiera i valori di B e H. Conviene moficare il programma in modo da avere una sola istruzione di Input. Per prima cosa cancelliamo l'istruzione 40.

Se l'utente vuole cancellare la linea 40, deve impostare:

D **E** **L** **E** **T** **E** **SPACE** **4** **0** **CR**

oppure

4 **0** **CR**

Per vedere il risultato di questo comando, l'utente deve impostare un altro comando LIST.

VIDEO	COMMENTI
LIST 10 REM RETTANGOLO1 20 INPUT "Base";B 30 IF B<=0 THEN 20 50 IF H<=0 THEN 40 60 LET AREA=A*H 70 PRINT "Area=";AREA;" B=";B;" H=";H 80 GOTO 20 90 END OK	In questa versione RETTANGOLO1 non può essere eseguito. L'utente deve correggere sia l'istruzione 20 (che richiede di introdurre un solo valore), che l'istruzione 50 (che fa riferimento a un numero di linea che non esiste più). Correggeremo il nostro programma nelle pagine successive.

DELETE (IMMEDIATO)

Cancella le linee di programma. L'M20 ritorna in Stato Comandi dopo che è stato eseguito un comando DELETE.

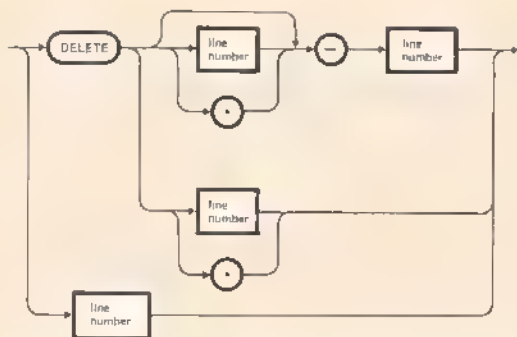



Figura 3-1 Comando DELETE

Esempi

SE l'utente imposta...	ALLORA...
DELETE . CR	la linea attuale viene cancellata
500 CR OPPURE DELETE 500 CR	la linea 500 viene cancellata
DELETE 100-200 CR	tutte le linee tra la 100 e la 200 comprese vengono cancellate
DELETE -400 CR	tutte le linee dall'inizio del programma fino alla linea 400 compresa vengono cancellate.

Nota: Se l'utente specifica con il comando DELETE dei numeri di linea che non esistono nel programma residente in memoria, il BASIC emette il messaggio "Illegal function call".

SOSTITUZIONE DI LINEE

Per modificare una linea l'utente può:

- sostituire l'intera linea, impostando il numero di linea e il suo nuovo contenuto, oppure

- modificare una parte di linea utilizzando l'Editor di linea.

Dapprima usiamo il primo metodo e sostituiamo la linea 20 e la linea 50 del programma RETTANGOLO1, impostando:

```
20 INPUT "Base e Altezza";B,H CR
50 IF H<=0 THEN 20 CR
```

e il programma sarà così listato:

VIDEO	COMMENTI
LIST 1Ø REM RETTANGOLO1 2Ø INPUT "Base e Altezza";B,H 3Ø IF B<=Ø THEN 2Ø 5Ø IF H<=Ø THEN 2Ø 6Ø LET AREA=B*H 7Ø PRINT "Area=";AREA;" B=";B;" H=";H 8Ø GOTO 2Ø 9Ø END Ok	Questa versione di RETTANGOLO1 è corretta. Comunque per terminare l'esecuzione, l'utente dovrà ancora impostare: **CTRL** **C**

Però non è pratico dover impostare **CTRL** **C** alla fine dell'esecuzione del programma, pertanto faremo alcune ulteriori modifiche. Possiamo sostituire alla linea 8Ø, le seguenti due linee:

1) INPUT "Ancora:SI=S,NO=N";X$
2) IF X$="S" THEN 2Ø

Per sostituire l'istruzione GOTO alla linea 8Ø, l'utente deve impostare:

8Ø INPUT "Ancora:SI=S,NO=N";X$ **CR**

Nota: X$ è una variabile stringa.

INSERIMENTO DI LINEE

Ora dobbiamo inserire l'istruzione 2) fra la linea 8Ø e la linea 9Ø. Possiamo scegliere 85 come numero di linea e impostare:

85 IF X$= "S" THEN 2Ø **CR**

Ora impostiamo un altro comando LIST:

VIDEO	COMMENTI
LIST 1Ø REM RETTANGOLO1 2Ø INPUT "Base e Altezza";B,H 3Ø IF B<=Ø THEN 2Ø 5Ø IF H<=Ø THEN 2Ø 6Ø LET AREA=B*H 7Ø PRINT "Area=";AREA;" B=";B;" H=";H 8Ø INPUT "Ancora:SI=S,NO=N";X$ 85 IF X$="S" THEN 2Ø 9Ø END Ok	Questa versione di RETTANGOLO1 non richiede che l'utente imposti CTRL C per arrestarne l'esecuzione. Però la sequenza dei numeri di linea non ha più un intervallo di 1Ø.

Quando l'utente lancia l'esecuzione del programma, ecco cosa succede: dopo aver calcolato l'area del rettangolo corrispondente ai valori introdotti della base e dell'altezza, il programma chiede all'utente se vuole continuare a introdurre altri valori. In caso affermativo, l'utente premerà S. L'istruzione 85 riporta allora il controllo all'istruzione 2Ø e si ripete la sequenza di calcolo. In caso negativo, l'utente premerà N. L'istruzione 85 non riporta in tal caso il controllo all'istruzione 2Ø, ma viene eseguita l'istruzione successiva che è un'istruzione END che arresta l'esecuzione del programma.

COME RINUMERARE LE LINEE

Come abbiamo visto, l'attuale numerazione delle linee di RETTANGOLO1 non ha più un intervallo di 1Ø. Ciò non ha importanza quando il programma è semplice, ma quando il programma è complesso e suscettibile di ulteriori modifiche, una numerazione delle linee casuale è sempre sconsigliabile.

Il comando RENUM consente di rinumerare le linee di un programma, iniziando ad esempio con 1Ø e con un intervallo di 1Ø tra le linee.
L'utente deve solo impostare:

R E N U M CR

Per vedere il risultato, l'utente deve impostare un altro comando LIST.

```
LIST
10 REM RETTANGOLO1
20 INPUT "Base e Altezza"; B,H
30 IF B<=0 THEN 20
40 IF H<=0 THEN 20
50 LET AREA=B*H
60 PRINT "Area=";AREA;" B=";B;" H=";H
70 INPUT "Ancora:SI=S,NO=N";X$
80 IF X$="S" THEN 20
90 END
Ok
```

MODIFICA AUTOMATICA DEI NUMERI DI LINEA RIFERITI

Quando un programma viene rinumerato tramite il comando RENUM, tutti i numeri di linea riferiti all'interno del programma vengono aggiornati in modo automatico. Per es., se un programma contiene l'istruzione GOTO 140 e la linea 140 viene rinumerata, il riferimento a questa linea nell'istruzione GOTO sarà aggiornato in modo automatico.

Regola Generale

Il comando RENUM modifica tutti i numeri di linea riferiti dopo GOTO, GOSUB, THEN, ELSE, ON...GOTO, ON...GOSUB e ERL, correntemente alla nuova numerazione delle linee.

Se in un programma si fa riferimento a linee non esistenti, il comando Renum fa comparire il messaggio seguente:

Undefined line XXXXX in YYYYY.

Il programma sarà rinumerato correttamente e i riferimenti a linee non esistenti rimarranno immutati.

RENUM (IMMEDIATO)

Modifica i numeri di linea del programma residente in memoria.

Figura 3-2 Comando RENUM

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO	VALORI DI DEFAULT
new line number	è il primo numero di linea della nuova numerazione	1Ø
old line number	è il primo numero di linea della precedente numerazione	se omesso corrisponde alla prima linea del programma
interval	è il nuovo intervallo tra i numeri di linea	1Ø

Esempi

SE l'utente imposta...	ALLORA...
RENUM **CR**	l'intero programma viene rinumerato. Il primo numero di linea nella new line number è 1Ø e viene assunto un intervallo di 1Ø (valori di default)
RENUM 1ØØ **CR**	L'intero programma viene rinumerato. Il primo numero di linea è 1ØØ e viene assunto un intervallo di 1Ø (valore di default).

RENUM 15Ø,,2Ø **CR**	L'intero programma viene rinumerato. Il primo numero di linea è 15Ø e viene assunto un intervallo di 2Ø

MODIFICA DI LINEE CON L'EDITOR DI LINEA

In Stato Editor è possibile modificare parti di una linea, senza dover riscrivere l'intera linea.

L'M20 entra in Stato Editor di linea se:

- l'utente imposta un comando EDIT, oppure

- viene riscontrato un errore di sintassi.

Dopo essere entrato in Stato Editor, l'M20 visualizza il numero della linea da modificare (che è seguito da uno spazio) e resta in attesa che l'utente imposti un comando dell'Editor. Questi comandi non vengono visualizzati quando l'utente li introduce, infatti, in Stato Editor, l'M20 interpreta i caratteri introdotti di volta in volta che l'utente li imposta, senza aspettare che l'utente prema **CR**.

EDIT (IMMEDIATO)

Il comando EDIT fa si che l'M20 entri in Stato Editor alla linea specificata.

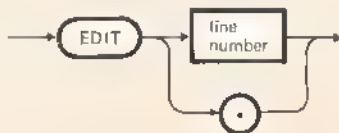



Figura 3-3 Comando EDIT

Esempi

SE l'utente imposta...	ALLORA l'M20 visualizza...
EDIT . CR	nn...n (entrando in Stato Editor alla linea attuale) In questo caso nn...n indica il numero della linea attuale
EDIT 3ØØ CR	3ØØ (entrando in Stato Editor alla linea specificata (quindi alla 3ØØ))

COMANDI DELL'EDITOR

La seguente tabella illustra i comandi dell'Editor che vengono raggruppati in classi.

CLASSE	COMANDO	SIGNIFICATO
inizio modifiche	L (Lista)	visualizza lo stato attuale della linea. Il numero della linea attuale viene visualizzato all'inizio della linea successiva.
	A (Again-Ancora)	ripristina la linea originaria senza visualizzarla. Il numero della linea attuale viene di nuovo visualizzato all'inizio della linea successiva.
spostamento cursore	n SPACE	visualizza gli n caratteri successivi e sposta il cursore a destra di una posizione.

	CTRL H (Backspace)	Cancella l'ultimo carattere della linea e sposta il cursore a sinistra di una posizione.
inserimento caratteri	I (Inserisce)	entra in Stato Inserimento all'attuale posizione del cursore. L'utente può inserire una stringa di caratteri e i caratteri inseriti vengono visualizzati. Per uscire dallo Stato Inserimento, l'utente deve impostare CTRL HOME
	X (Extended Line-Inserisce a fine linea)	visualizza l'ultima parte della linea, sposta il cursore alla fine della linea e entra in Stato Inserimento
	CTRL HOME	esce dallo Stato Inserimento, ma rimane in Stato Editor. Se l'utente preme CR, esce sia dallo Stato Inserimento che dallo Stato Editor
Cancella caratteri	D (Delete-Cancella un carattere)	cancella il carattere successivo che viene visualizzato tra due barre rovesce (\ ) e il cursore viene posizionato alla sua destra

	n **D** (Delete-Cancella n caratteri)	cancella i successivi caratteri. I caratteri cancellati sono visualizzati tra due barre rovesce (\ ); il cursore viene posizionato alla destra dell'ultimo carattere cancellato. Se la linea ha meno di n caratteri alla destra del cursore, **n** **D** cancella l'ultima parte della linea
	H (Hack-Tronca)	cancella l'ultima parte della linea e entra in Stato Inserimento
ricerca caratteri	**S** **x** (Search-Ricerca la prima occorrenza di x)	ricerca la prima occorrenza di "x" nella linea (dove "x" è un qualsiasi carattere stampabile ASCII) e posiziona il cursore subito prima di questo carattere. Il carattere alla posizione attuale del cursore non viene incluso nella ricerca. Se il carattere non viene trovato nella linea, il cursore viene portato a fine linea e tutti i caratteri esaminati durante la ricerca vengono visualizzati.

	n **S** **x** (Search-Ricerca l'n-esima occorrenza di x)	ha la stessa funzione del comando precedente, ma ricerca l'n-esima occorrenza anzichè la prima.
	K **x** (Delete-Cancella fino alla prima occorrenza di x)	ha una funzione simile al comando **S** **x**, ad esclusione del fatto che tutti i caratteri esaminati durante la ricerca vengono cancellati. Il cursore viene posizionato subito prima di "x" e i caratteri cancellati vengono racchiusi tra due barre rovesce (\ ).
	n **K** **x** (Delete-Cancella fino all'n-esima occorrenza di x)	ha la stessa funzione del comando precedente, ma ricerca l'n-esima occorrenza, anzichè la prima.
sostituzione caratteri	**C** **x** (Cambia un carattere)	cambia in "x" il carattere successivo a quello della posizione del cursore.
	n **C** **x1** **x2** ... **xn** (Cambia una stringa di n caratteri)	cambia i successivi n caratteri nella stringa specificata (cioè impostata dopo **C**). Quando l'utente ha impostato una stringa di n caratteri, l'M20 ritorna in Stato Editor uscendo dallo Stato Sostituzione caratteri.

uscita dall'Editor	CR	visualizza la linea modificata e ritorna allo Stato Comandi
	E (Exit-Uscita)	ha la stessa funzione di CR ma l'ultima parte della linea non viene visualizzata.
	Q (Quit-Uscita con ripristino)	torna allo Stato Comandi e cancella tutti i cambiamenti fatti alla linea.

Esempi

La seguente tabella fornisce alcuni esempi di uso dei comandi dell'Editor.

Si noti che la posizione del cursore viene evidenziata come indicato (_) quando l'M20 é in stato Editor.

	SE l'utente imposta...	ALLORA l'M20 visualizza...
1	E Q I I SPACE 5 Ø Ø CR	5ØØ _
2	L	5ØØ FOR I=1 TO 15 STEP 2 5ØØ _
3	SPACE (6 volte)	5ØØ FOR I=_
4	C 2	5ØØ FOR I=2_
5	SPACE (5 volte)	5ØØ FOR I=2 TO 1_
6	C 6	5ØØ FOR I=2 TO 16_
7	CR	5ØØ FOR I=2 TO 16 STEP 2 █
1	E D I I SPACE 5 1 Ø CR	51Ø _

2	L	51Ø LET A(I)=I*SIN(X) 51Ø _
3	SPACE (11 volte)	51Ø LET A(I)=I*_
4	3 C L O S	51Ø LET A(I)=I*COS_
5	X : P R I N T SPACE A (I) CR	51Ø LET A(I)=I*COS(X):PRINT A(I) ▮
1	E D I T SPACE . CR	51Ø _
2	4 D	51Ø \LET \ _
3	SPACE (11 volte)	51Ø \LET \ A(I)=I*COS(_
4	I Y + [illegible] [illegible]	51Ø \LET \ A(I)=I*COS(Y+ _
5	SPACE (9 volte)	51Ø \LET \ A(I)=I*COS(Y+X):PRINT _
6	4 C I , X ; CR	51Ø \LET \ A(I)=I*COS(Y+X): PRINT I,X; ▮
7	L I S T SPACE . CR	51Ø A(I)=I*COS(Y+X):PRINT I,X; ▮

ESAME DEI VALORI ATTUALI DELLE VARIABILI

Quando l'utente modifica una linea di programma tramite l'Editor di linea, tutte le variabili numeriche vengono azzerate e le variabili stringa vengono inizializzate con il valore "stringa nulla" (""). Inoltre vengono chiusi tutti i file dati. Se il BASIC rivela un errore sintattico durante l'esecuzione di un programma, passa automaticamente in Stato Editor. Prima di modificare una linea, l'utente potrebbe aver la necessità di sapere i valori attuali delle variabili. In questo caso dovrà premere il tasto **Q**, come primo comando di Editor. Questo comando fa passare dallo Stato Editor allo Stato Comandi, dove è possibile esaminare i valori delle variabili. Qualsiasi altro comando di Editor (premendo il tasto **E** oppure **CR** ecc.) azzererà tutte le variabili numeriche e inizializzerà tutte le variabili stringa con il valore "stringa nulla" (""), rendendo perciò impossibile l'esame dei valori delle variabili.

COME RINOMINARE UN FILE

L'utente può cambiare il nome di un file programma o di un file dati residenti su disco tramite il comando NAME, purchè il disco o il file non siano protetti da scrittura. Il volume selezionato deve includere il nome da sostituire, e non deve includere il nuovo nome del file. Dopo che un comando NAME è stato eseguito, il file continua a esistere sullo stesso disco e nella stessa area di disco, ma ha il nuovo nome. Le password di file e di volume (se esistevano), non vengono modificate. L'utente deve in tal caso specificare la password di file e il volume deve essere abilitato (altrimenti l'utente deve specificare anche la password di volume).

NAME (PROGRAMMA/IMMEDIATO)

Cambia il nome di un file su disco.

Figura 3-4 Comando NAME

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file identifier	Può essere una costante o una variabile stringa e specifica il file programma o dati il cui nome deve essere modificato
file name	Può essere una costante o una variabile stringa e specifica il nuovo nome del file

Esempi

Si suppone che nè il volume nè il file siano protetti da scrittura e che il volume sia abilitato.

SE l'utente imposta...	ALLORA...
NAME "1:FR1" AS "FR2" CR	il nome di file FR1 viene cambiato in FR2. Il file risiede sul dischetto inserito nella unità 1. FR1 non ha password.
NAME "VOL1:ACC/PACC" AS "ACC1" CR	Il nome di file ACC viene cambiato in ACC1. Il file risiede sul dischetto VOL1 che può essere inserito sia nella prima che nella seconda unità. La password del file resta PACC.

COME CANCELLARE UN FILE

Tramite il comando KILL l'utente può facilmente cancellare i file programma oppure i file dati registrati su disco, purchè il disco non sia protetto da scrittura. Il nome di un file che è stato cancellato, può essere riutilizzato per identificare un nuovo file.

L'utente deve specificare la password del file (se esiste), e il volume deve essere abilitato (altrimenti l'utente deve specificarne la password).

KILL (PROGRAMMA/IMMEDIATO)

Cancella un file programma o un file dati registrati su disco.

Figura 3-5 Comando KILL

Dove

File identifier può essere una costante o una variabile stringa e specifica il file da cancellare.

Esempi

Si suppone che il volume non sia protetto da scrittura e che sia abilitato.

SE l'utente imposta...	ALLORA...
KILL "Business.B" CR	il file Business.B viene cancellato. Esso risiede sull'unità selezionata per ultima. Il file non ha password.
KILL "1:Business.B" CR	il file Business.B viene cancellato. Esso risiede sul dischetto inserito nell'unità 1. Il file non ha password.
KILL "NUMbers/PNUMØ1" CR	il file NUMbers con password PNUMØ1 viene cancellato. Esso risiede sull'unità selezionata per ultima.

COME FARE IL MERGE DI DUE PROGRAMMI

Il comando MERGE permette di inserire nel programma in memoria un altro programma residente su disco in formato ASCII, in modo da ottenere un unico programma. Il comando MERGE ha una funzione simile al comando LOAD, ma il programma residente in memoria non viene cancellato al momento del trasferimento in memoria del programma su disco. L'operazione di MERGE comporta che le linee del programma su disco vengano inserite (secondo la sequenza del numero di linea) nel programma residente in memoria. Se, in questo processo di inserimento, due linee hanno lo stesso numero di linee, la linea del programma su disco sostituisce quella in memoria. Il comando MERGE deve specificare la password di file, se il programma su disco ha una password e il disco deve essere abilitato (o l'utente deve specificarne la password).

L'operazione di MERGE può ad esempio essere utile per inglobare subroutine standard in un programma.

E' buona norma di programmazione fare l'operazione di MERGE tra un programma e subroutine con numeri di linea più alti del massimo numero di linee del programma al fine di ridurre il tempo dell'operazione di MERGE e di lasciare la possibilità di estendere, eventualmente, il programma.

MERGE (PROGRAMMA/IMMEDIATO)

Esegue l'operazione di MERGE tra il programma in memoria e il programma registrato su disco (in formato ASCII).

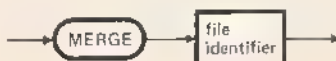



Figura 3-6 Comando MERGE

Dove

Il file identifier può essere una costante o una variabile stringa e specifica un file programma in formato ASCII, cioè registrato con l'opzione A.

Esempi

VIDEO	COMMENTI
MERGE "1:Fnew/FnewPASS" CR	si fa il MERGE tra il programma Fnew con la password FnewPASS e il programma in memoria. Nota: Il dischetto è inserito nella unità 1 ed è abilitato.

MERGE "VØØ1/VPØØ1:FØØ1/PØØ1" CR	si fa il MERGE tra il programma FØØ1 con la password PØØ1 e il programma in memoria. Nota: In questo caso il disco VØØ1 viene abilitato dal comando MERGE con l'uso della password VPØØ1.

Nota
MERGE chiude tutti i file dati eventualmente aperti dal programma residente in memoria e non ancora chiusi, azzera le variabili numeriche e inizializza le variabili stringa col valore stringa nulla.

COME LISTARE I NOMI DEI FILE REGISTRATI SU DISCO

Se l'utente non ricorda i nomi dei file programma e/o dati registrati su disco, può usare il comando FILES.

Questo comando può essere usato sia specificando un identificatore di volume che un identificatore di file.

Quando l'utente specifica un identificatore di volume, tutti i file registrati su quel disco vengono listati (sia che abbiano o no una password).

Per eseguire un comando FILES non è necessario nè conoscere la password di volume nè che il disco venga abilitato.

Quando l'utente specifica invece un identificatore di file, soltanto questo file viene listato e l'utente può anche non specificarne la password (se esiste).

La stessa funzione svolta dal comando FILES può essere realizzata in PCOS tramite il comando VQUICK.

Nota: Il comando FILES non lista la password.

Il comando FILES permette di visualizzare:

- l'unità disco, dove il disco stesso è inserito.

- il nome del disco (se esiste)

- lo spazio libero su disco in settori (dove un settore è pari a 256 byte).

- il nome di ogni file su disco, oppure del file specificato, oppure il nome dei file selezionati se l'utente utilizza i caratteri "Jolly" (? o *) nell'identificatore di file. Si tenga presente che il punto interrogativo (?) identifica un carattere qualsiasi e che l'asterisco (*) identifica una qualsiasi sequenza di caratteri.

FILES (PROGRAMMA/IMMEDIATO)

Lista i file registrati sul disco specificato.

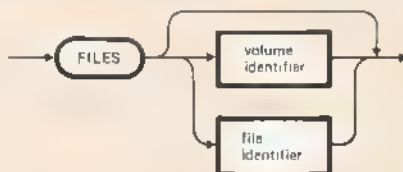



Figura 3-7 Comando FILES

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
volume identifier	può essere una costante o una variabile stringa e specifica il disco di cui si richiede di listare i file
file identifier	può essere una costante o una variabile stringa e specifica il file di cui si richiede di visualizzare il nome e l'estensione (in byte)

Esempi

SE l'utente imposta...	ALLORA...
FILES CR	viene visualizzato il nome di ogni file residente sul disco inserito nell'unità selezionata per ultima.
FILES "Ø:" CR	viene visualizzato il nome di ogni file residente sul disco inserito nella unità Ø.
FILES "1Ø:" CR	viene visualizzato il nome di ogni file residente su hard disk.
FILES "MYVOL:" CR	viene visualizzato il nome di ogni file residente sul disco MYVOL. Esso può essere inserito sia nella prima che nella seconda unità.
FILES "MYVOL/MYPASS:" CR	viene visualizzato il nome di ogni file residente sul disco MYVOL che ha la password MYPASS. Esso può essere inserito sia nella prima che nella seconda unità. L'utente può specificare o meno la password di volume: ciò non é rilevante per l'esecuzione di questo comando.
FILES "MYFILE" CR	viene visualizzato il nome del file MYFILE, che è residente sul disco inserito nell'ultima unità selezionata.
FILES "1:*.cmd" CR	viene visualizzato il nome di tutti i file che hanno l'estensione 'cmd' e che risiedono sul dischetto inserito nell'unità 1.
FILES "Ø:V???" CR	viene visualizzato il nome di tutti i file che risiedono sul dischetto inserito nell'unità Ø e che soddisfano alle seguenti condizioni: - il nome inizia con la lettera 'V' - il nome é composto da quattro caratteri

4. I DATI

SOMMARIO

In questo capitolo prenderemo in considerazione come il linguaggio BASIC tratta i dati. Esamineremo le costanti e le variabili, la rappresentazione dei numeri, le conversioni numeriche e le matrici.

INDICE

COSTANTI E VARIABILI

I dati che un programma BASIC può prendere in considerazione, possono essere delle costanti o delle variabili.

COSTANTI

Se in un'istruzione BASIC compare un numero (ad es. 15, -2, 3.41 ecc.), o una stringa di caratteri (ad es. "AAA.b1", "Cursor***"), questo numero o questa stringa sono considerati costanti. Ciò significa che i loro valori non variano durante l'esecuzione del programma.

VARIABILI

Una variabile è un dato al quale è stato associato un nome. Il valore del dato può quindi cambiare durante l'esecuzione del programma.

Per es., la formula che calcola l'area di un cerchio:

3.141592*R∧2

utilizza la variabile R, che rappresenta qualsiasi valore del raggio, e riserva una posizione di memoria per questo valore.

Nota: Il simbolo ∧ è un operatore che indica che R deve essere elevato alla potenza specificata (nel nostro caso 2).

NOMI DELLE VARIABILI

L'identificatore (o nome) di una variabile non può essere più lungo di 40 caratteri. I caratteri ammessi possono essere lettere e numeri. Il punto (.) è pure ammesso. Il primo carattere deve essere una lettera, e l'ultimo può essere sia una lettera, sia un numero, sia un punto, sia un carattere di dichiarazione di tipo (%,!,#,$). Il significato di un carattere di dichiarazione di tipo verrà spiegato più avanti in questo stesso capitolo.
Le lettere minuscole, eventualmente presenti in un identificatore di variabile, sono considerate equivalenti alle corrispondenti lettere maiuscole e vengono convertite in lettere maiuscole quando si lista il programma.

Esempi di nome di variabile sono:

STUDENTE   A1   CCØ1.CLASSE   ACCONTO#   A$ STRINGA.

Parole Riservate

Una parola riservata (una parola chiave, il nome di un comando o di una funzione) non può essere usata come identificatore di una variabile, ma il BASIC consente di inserire parole riservate all'interno, all'inizio o alla fine di un identificatore di variabile. Per es.:

```
10 PERFORMACE = 105.3
20 SINGLE = 1371.2
```

sono linee di programma corrette, anche se PERFORMANCE contiene la parola chiave FOR, e SINGLE inizia con il nome della funzione SIN.

RAPPRESENTAZIONE DEI NUMERI

L'uomo è abituato a pensare i numeri in base 10 (rappresentazione decimale). Un computer considera invece i numeri come stringhe di bit (dove ogni bit può avere il valore 0 oppure 1), ed esegue la maggior parte delle sue operazioni trattando i numeri in base 2 (rappresentazione binaria).

Questo paragrafo traccia una panoramica dei concetti di base della rappresentazione dei numeri in base 2, in base 16 e in base 8.

RAPPRESENTAZIONE BINARIA

Prima di parlare della rappresentazione binaria, esaminiamo brevemente la struttura della rappresentazione decimale. La rappresentazione decimale usa le cifre 0,1,2,...9. Ad esempio, prendiamo in considerazione il numero:

205

Vediamo che le sue cifre hanno un valore di posizione associato alla potenza del 10. Infatti questo numero può essere pensato come:

$$(2 \times 10^2) + (0 \times 10^1) + (5 \times 10^0)$$

Il concetto di valore di posizione esiste anche nella rappresentazione binaria. L'unica differenza é che vengono prese in considerazione le potenze del 2, anziché quelle del 10. Il numero 205 in base 2 é:

11ØØ11Ø1

Si noti che in base 2, le cifre ammesse sono soltanto "1" e "Ø". Quindi la predetta rappresentazione binaria significa:

$(1 \times 2^7) + (1 \times 2^6) + (Ø \times 2^5) + (Ø \times 2^4) + (1 \times 2^3) + (1 \times 2^2) +$

$(Ø \times 2^1) + (1 \times 2^Ø)$

cioè:

128 + 64 + 8 + 4 + 1 = 2Ø5

Una cifra binaria viene denominata bit. Un bit può essere "1" o "Ø".

Byte

Le stringhe di bit vengono solitamente suddivise in gruppi di 8 bit. Un gruppo di 8 bit, considerato come una singola unità di informazione, viene denominata "byte".

I bit di un byte sono numerati da Ø (il primo a destra, cioé il meno significativo) a 7 (il primo a sinistra, cioé il più significativo).

In base a questa convenzione, il numero del bit e la potenza del 2 che rappresenta sono identici. La seguente tabella fa vedere il numero di posizione di un bit nell'ambito del byte e il suo valore corrispondente.

NUMERO DI POSIZIONE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT Ø
Significato	2^7	2^6	2^5	2^4	2^3	2^2	2^1	$2^Ø$
Valore	128	64	32	16	8	4	2	1

Tabella 4 - 1

BASE 1Ø	BASE 2
Ø	Ø
12	11ØØ
27	11Ø11
149	1ØØ1Ø1Ø1
255	11111111

Tabella 4 - 2 Esempi di conversione

Word

L'M20 elabora 16 bit (2byte) per volta. Questa quantità é detta "word". Il numero di bit in una word può variare da macchina a macchina. I bit di una word sono numerati da Ø (quello più a destra, cioé il meno significativo) a 15 (quello più a sinistra, cioé il più significativo).

Un'altra caratteristica di una word nel sistema M20 é che viene usata la rappresentazione "completamento a due". Questa rappresentazione consente di memorizzare in una word sia numeri positivi che negativi:

SE un numero intero é..	ALLORA	E la word é ...
positivo	il bit 15 é Ø	un numero positivo rappresentato in binario
negativo	il bit 15 é 1	un numero negativo rappresentato in "completamento a due".

Per trovare il valore assoluto di un numero negativo, bisogna invertire tutti i suoi bit e aggiungere 1.

Per esempio·

```
1111ØØ11ØØ11ØØØ    valore originario (negativo)
```

Invertendo tutti i bit

```
ØØØØØ11ØØ11ØØ111    valore invertito
```

Aggiungendo 1

```
ØØØØØ11ØØ11Ø1ØØØ    Valore assoluto (valore invertito + 1)
```

Sicché il valore della configurazione di bit sopra specificata é:

-164Ø

RAPPRESENTAZIONE ESADECIMALE E OTTALE

Abbiamo visto che è possibile rappresentare: numeri in base 10 (rappresentazione decimale) e in base 2 (rappresentazione binaria). IL BASIC permette anche di rappresentare: numeri in base 8 (rappresentazione ottale) e in base 16 (rappresentazione esadecimale).

Sebbene sia spesso conveniente trattare i numeri in base 2, è difficile per l'utente scriverli e leggerli in questa rappresentazione. Per questa ragione l'utente preferisce convertirli in base otto o in base sedici.

- Base 8, conosciuta come "rappresentazione ottale", usa una cifra ottale per tre cifre binarie

- Base 16, conosciuta come "rappresentazione esadecimale", usa una cifra esadecimale per 4 cifre binarie.

La seguente tabella illustra la rappresentazione in base 10 (decimale), in base 2 (binaria), in base 8 (ottale), e in base 16 (esadecimale) dei numeri da 0 a 16.

DECIMALE	BINARIA	OTTALE	DECIMALE	BINARIA	ESADECIMALE
0	000	0	0	0000	0
1	001	1	1	0001	1
2	010	2	2	0010	2
3	011	3	3	0011	3
4	100	4	4	0100	4
5	101	5	5	0101	5
6	110	6	6	0110	6
7	111	7	7	0111	7
8	1000	10	8	1000	8
9	1001	11	9	1001	9
10	1010	12	10	1010	A
11	1011	13	11	1011	B
12	1100	14	12	1100	C
13	1101	15	13	1101	D
14	1110	16	14	1110	E
15	1111	17	15	1111	F
16	10000	20	16	10000	10

Tabella 4 - 3

CLASSIFICAZIONE DELLE COSTANTI

Il modo in cui il BASIC memorizza un dato determina:

- lo spazio in memoria che il dato occupa (in byte)

- la velocità con cui il BASIC elabora il dato.

DATI NUMERICI

L'utente può far sì che il BASIC memorizzi i numeri del proprio programma come:

- numeri interi (massima velocità di elaborazione ma limitato campo di definizione)

- numeri in semplice precisione (di uso generale)

- numeri in doppia precisione (massima precisione ma elaborazione più lenta).

	NUMERI INTERI	NUMERI IN SEMPLICE PRECISIONE	NUMERI IN DOPPIA PRECISIONE
Spazio in memoria (in byte)	2	4	8
Campo di definizione	Da -32768 a 32767	Da $\pm 10^{-38}$ a $\pm 10^{38}$	Da $\pm 10^{-308}$ a $\pm 10^{308}$
Cifre significative	Massimo 5	Massimo 7	Massimo 16
Cifre visualizzate (PRINT/LPRINT)	Massimo 5	Massimo 6 (con arrotondamento)	Massimo 15 (con arrotondamento)

Tabella 4 - 4

Nota: Gli zeri non significativi non vengono visualizzati. Per esempio il valore 3.41ØØØØ in semplice precisione verrà visualizzato come 3.41.

DATI STRINGA

Le stringhe (sequenze di caratteri ASCII) sono utilizzate per memorizzare informazioni non numeriche, come nomi, indirizzi, codici,ecc. Per esempio la costante:

"FORD ,RENAULT"

è una costante stringa di 13 caratteri. Ogni carattere della stringa (compreso lo spazio) è memorizzato in un byte e corrisponde a un codice della tabella ASCII. Il BASIC memorizza la suddetta costante stringa come segue:

Carattere ASCII	F	O	R	D		,	R	E	N	A	U	L	T
Codice Esadecimale	46	4F	52	44	2Ø	2C	52	45	4E	41	55	4C	54

Tabella 4 - 5

La lunghezza massima di una stringa è di 255 caratteri. Una stringa con lunghezza zero è detta stringa "nulla", ed è rappresentata da una coppia di virgolette (""). Il BASIC memorizza le stringhe in modo dinamico, cioè lo spazio in memoria riservato ad una stringa può variare durante l'esecuzione del programma da Ø a 255 byte.

	STRINGA NULLA	STRINGA DI n CARATTERI	STRINGA DI MASSIMA LUNGHEZZA
Spazio in memoria (in byte)	Ø	n	255
Campo di definizione	-	Qualsiasi stringa di caratteri stampabili ASCII, spazio compreso	

Tabella 4 - 6

DETERMINAZIONE DEL TIPO DI UNA COSTANTE

SE...	ALLORA...	ESEMPI
il valore è compreso tra due virgolette	è una stringa	"NO" "YES" "Circle" ""(stringa nulla)
il valore non è compreso tra due virgolette	è un numero Nota: Un'eccezione a questa regola si ha quando l' operatore introduce un dato in risposta a una istruzione di INPUT o LINE INPUT, e nelle istruzioni DATA	521 -15 3.7345E-2 43#
un numero è intero ed è compreso tra -32768 e 32767	è una costante intera	1Ø24 721 -32768
il valore ha il prefisso &H, ed è composto dai numeri da Ø a 9 e dalle lettere da A a F (compreso tra Ø e FFFF)	è una costante esadecimale Nota: una costante esadecimale può essere considerata come una rappresentazione alternativa della corrispondente costante intera	&H2ØFØ &HF1 &H35 &HFE98 &HFFFF &HØ
il valore ha il prefisso &O o & ed è composto dai numeri da Ø a 7 (compreso tra Ø e 177777)	è una costante ottale Nota: una costante ottale può essere considerata una rappresentazione alternativa della corrispondente costante intera	&O7Ø &O44 &71175

un numero non è un numero intero e non contiene più di 7 cifre	è una costante in semplice precisione	-2.3 32768 45.314 -65ØØØ
un numero contiene più di 7 cifre	è una costante in doppia precisione	52174593 -54.397124 8.799999999

CARATTERI DI DICHIARAZIONE TIPO

L'utente può forzare il tipo di una costante aggiungendo alla fine di una costante numerica i seguenti caratteri di dichiarazione:

CARATTERI DI DICHIARAZIONE	SIGNIFICATO	ESEMPI
!	dichiara un numero in semplice precisione	5.7211Ø333! la costante è in semplice precisione e solo le prime 7 cifre (per es. 5.7211Ø3) vengono memorizzate.
E	semplice precisione "floating point". E indica che la costante deve essere moltiplicata per una potenza del 1Ø, indicata dopo la E	7.31E4 significa $7.31 \times 1Ø^{4}$ cioè 731ØØ
#	dichiara un numero in doppia precisione	4# 5.21#
D	doppia precisione "floating point". D indica che la costante deve essere moltiplicata per una potenza del 1Ø, indicata dopo la D	7.2D-3 significa $7.2 \times 1Ø^{-3}$ cioè Ø.ØØ72

CLASSIFICAZIONE DELLE VARIABILI

Ogni identificatore di variabile che appare in un programma viene classificato dal BASIC o come una stringa, o come un numero intero, o come un numero in semplice o doppia precisione.

Se non viene specificato altrimenti, il BASIC classifica tutte le variabili numeriche in semplice precisione. Per esempio, se questa è la prima linea del nostro programma:

```
1Ø X1=3.5
```

il BASIC classifica X1 come una variabile in semplice precisione.

Tuttavia, l'utente può attribuire tipi diversi alle variabili sia usando istruzioni di definizione di tipo sia scrivendo un carattere particolare, atto a dichiarare il tipo, alla fine dell'identificatore di variabile.

DEFINT/DEFSNG/DEFDBL/DEFSTR (PROGRAMMA/IMMEDIATO)

Esistono in BASIC quattro istruzioni di definizione di tipo.

Un'istruzione di definizione di tipo dichiara il tipo di tutte le variabili il cui nome inizia con una lettera ben specificata.

Queste istruzioni vengono inserite di solito all'inizio del programma e devono precedere l'uso delle variabili di cui definiscono il tipo.

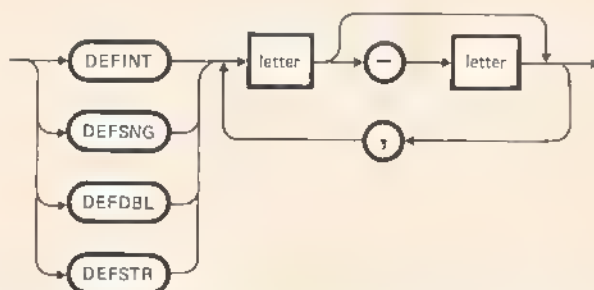



Figura 4-1 Istruzioni di Definizione di Tipo

Valori di Default

Se non specificato altrimenti, tutte le variabili vengono assunte in semplice precisione.

Esempi

SE l'utente imposta...	ALLORA...
1Ø DEFINT A-Z CR	tutte le variabili del programma saranno numeri interi
1Ø DEFDBL D CR	tutte le variabili del programma che iniziano con la lettera D saranno in doppia precisione
1Ø DEFSTR S,U-W CR	tutte le variabili del programma che iniziano con le lettere S, U, V e W saranno variabili stringa

CARATTERI DI DICHIARAZIONE DI TIPO

Come già abbiamo visto per le costanti, l'utente può forzare il tipo di una variabile aggiungendo alla fine del nome un carattere di dichiarazione. Per le variabili abbiamo quattro caratteri di dichiarazione di tipo:

CARATTERE DI DICHIARAZIONE	SIGNIFICATO	ESEMPI
%	variabile intera	A% STEP% INCREMENT% sono tutte variabili intere, indipendentemente da eventuali istruzioni di definizione di tipo per le variabili che iniziano con A, S e I.

!	variabile in semplice precisione	SPEED! SPACE! TIME! sono tutte variabili in semplice precisione, indipendentemente da eventuali istruzioni di definizione di tipo per le variabili che iniziano con le lettere S e T.
#	variabile in doppia precisione	TOTAL# SUBTOTAL# X1# sono tutte variabili in doppia precisione, indipendentemente da eventuali istruzioni di definizione di tipo per le variabili che iniziano con le lettere T, S e X.
$	variabile stringa	A$ B1$ NAME.CLASS$ sono tutte variabili stringa, indipendentemente da eventuali istruzioni di tipo per le variabili che iniziano con le lettere A, B e N.

CONVERSIONI NUMERICHE

Spesso in un programma o in una linea ad esecuzione immediata si assegna una costante di un certo tipo ad una variabile di tipo diverso.
Per esempio, se si imposta:

```
I% = 5.31 CR
```

il BASIC arrotonderà per prima cosa la costante 5.31 (in semplice precisione) all'intero più vicino. Questo numero verrà poi assegnato alla variabile intera I%. Perciò il valore della variabile I% sarà 5.

Si può anche assegnare a una variabile di un certo tipo una variabile di tipo diverso, come ad esempio:

```
SCALE! = B% CR
SECONDS! = C1# CR
BOX# = W% CR
```

Le procedure di conversione sono elencate nelle pagine seguenti.

DA PRECISIONE SEMPLICE O DOPPIA A INTERO

Il BASIC converte il valore in semplice o doppia precisione a un intero arrotondando la parte frazionaria.

Nota: Il valore ottenuto deve essere maggiore o uguale a -32763, e minore di 32767, altrimenti si ha un errore di Overflow.

Esempi

VIDEO	COMMENTI
C% = -15.1 Ok ?C% -15 Ok	il valore -15 è assegnato alla variabile C%
C% = 4.1E2 Ok ?C% 410 Ok	il valore 410 è assegnato alla variabile C%
C% = 47.8 Ok ?C% 48 Ok	il valore 48 è assegnato alla variabile C%

VIDEO	COMMENTI
C% = 7.21473D-3 Ok ?C% Ø Ok	il valore Ø è assegnato alla variabile C%
C% = -32768.5 Overflow Ok	errore di Overflow

DA INTERO A PRECISIONE SEMPLICE O DOPPIA

In questo caso la conversione non comporta errori, e il valore convertito corrisponde al valore originario esteso con zeri alla destra del punto decimale.

VIDEO	COMMENTI
S! = 326 Ok ?S! 326 Ok	il numero 326 è memorizzato nella variabile S! come 326.ØØØØ, ma visualizzato come 326.
D# = 326 Ok ?D# 326 Ok	il numero 326 è memorizzato nella variabile D# come 326.ØØØØØØØØØØØØØ, ma viene visualizzato come 326.

DA SEMPLICE A DOPPIA PRECISIONE

Il BASIC aggiunge degli zeri alla fine di un numero in semplice precisione.

Se il valore originario:

- ha una rappresentazione binaria esatta, la conversione non comporta errori

- non ha una rappresentazione binaria esatta, un errore aritmetico viene introdotto al momento della conversione.

Esempi

VIDEO	COMMENTI
B# = 1.5 Ok ?B# 1.5 Ok	Quando si introduce B# = 1.5, viene assegnato il valore 1.5ØØØØØØØØØØØØØØ alla variabile B# , ma questo valore è visualizzato soltanto come 1.5 Nota: 1.5 ha una rappresentazione binaria esatta
C# = 1.3 Ok ?C# 1.29999995231628 Ok	Quando si introduce C# = 1.3, viene assegnato il valore 1.29999995231628Ø alla variabile C# , ma questo valore è visualizzato soltanto come 1.29999995231628 Nota: 1.3 non ha una rappresentazione binaria esatta.

Note

Per evitare una perdita di precisione nei calcoli è buona norma di programmazione evitare ove possibile conversioni da semplice a doppia precisione nei programmi BASIC.

Ad esempio, se si vuole assegnare un valore costante a una variabile in doppia precisione, conviene usare una costante in doppia precisione.
Per esempio:

B #= 1.3#   B #= 1.3D

memorizzano entrambe 1.3 nella variabile B #.

Quando il valore in singola precisione è memorizzato in una variabile, è allora necessario (per evitare una perdita di precisione) convertire prima la variabile in semplice precisione a un valore stringa con la funzione STR$ (vedi Capitolo 9) e quindi convertire di nuovo la stringa risultante in un numero con la funzione VAL (vedi Capitolo 9).

VIDEO	COMMENTI
LIST 10 B!=1.3 20 B# =B! 30 PRINT B# Ok RUN 1.29999995231628 Ok	Questo programma visualizza il valore di B# con una perdita di precisione
LIST 10 B!=1.3 20 B# =VAL(STR$(B!)) 30 PRINT B# Ok RUN 1.3	Questo programma visualizza il valore di B# senza perdite di precisione

DA DOPPIA A SEMPLICE PRECISIONE

Questa operazione consiste nel convertire un numero che ha al massimo 16 cifre significative in un numero che non ne ha più di 7.

Saranno prese in considerazione soltanto le prime sette cifre del valore originario con arrotondamento dell'ultima cifra.

Prima di visualizzare o stampare tale numero il BASIC lo riduce, sempre con arrotondamento a 6 cifre.

Nota: Se il valore in doppia precisione è fuori dal campo di definizione dei valori in semplice precisione, si ha un errore di Overflow.

Esempio

VIDEO	COMMENTI
P! = 2.03999996 Ok ?P! 2.04 Ok	il valore 2.040000 viene assegnato alla variabile P!, ma questo valore viene visualizzato soltanto come 2.04

CONVERSIONI ILLECITE

L'utente non può convertire valori numerici in valori stringa, o viceversa, mediante un'istruzione di assegnazione. Per esempio:

C$ = 321.7

è una conversione che non è ammessa. (In questi casi le funzioni STR$ e VAL permettono di realizzare queste conversioni. Vedere Capitolo 9).

VARIABILI CON INDICE E MATRICI

Come predetto (Vedere Capitolo 1) una variabile può essere una "variabile semplice" oppure una "variabile con indice", cioè un elemento di "matrice" ("array").

Una "matrice" è un insieme di variabili dello stesso tipo, aventi tutte lo stesso nome ma distinguibili tramite il valore di uno o più indici indicati tra parentesi dopo il nome. Ad esempio, se A è il nome di una matrice a una dimensione, A(Ø) è il suo primo elemento, A(1) il secondo ecc.(supponendo che il valore minimo degli indici sia Ø).

Il valore di un indice deve essere un intero positivo, ma è possibile usare un'espressione numerica per esprimere il valore di un indice. Se il valore dell'espressione non fosse intero viene arrotondato automaticamente all'intero più vicino.

Una matrice può avere un numero qualsiasi di dimensioni. Una matrice a una dimensione può essere pensata come una lista di dati. Può avere più righe, ma una sola colonna. Una matrice a due dimensioni può essere passata come una tabella di dati. Può avere più righe e più colonne.

Per definire una matrice, l'utente deve:

- attribuire un nome alla matrice (le stesse regole specificate per i nomi delle variabili semplici, sono anche valide per i nomi delle matrici)
- stabilire il valore massimo e il valore minimo degli indici.

Per fare ciò, l'utente deve scrivere un'istruzione DIM ed eventualmente un'istruzione OPTION BASE. Se l'utente specifica nel suo programma:

```
1Ø OPTION BASE 1
```

il valore minimo degli indici di tutte le matrici è 1.

Se, invece, nel programma non è presente alcuna istruzione OPTION BASE (o se è presente l'istruzione OPTION BASE Ø), il valore minimo degli indici di tutte le matrici è Ø.

E' anche possibile ridefinire una matrice tramite una nuova istruzione DIM, preceduta da una istruzione ERASE (che verrà spiegata in dettaglio nel seguito).

MATRICI A UNA DIMENSIONE

Supponiamo di avere la seguente lista di numeri:

17, -9, 32, 1Ø5, -48

Se definiamo una matrice numerica a una dimensione V, possiamo memorizzare tutti i valori della lista come elementi di questa matrice e possiamo accedere a ciascun elemento tramite il valore del suo indice.

Matrice V

Elemento	Contenuto
V(Ø)	17
V(1)	-9
V(2)	32
V(3)	1Ø5
V(4)	-48

Ogni elemento della matrice V è individuato dal suo indice. Per esempio, il valore di V(1) è -9, e il valore di V(3) è 1Ø5. L'indice identifica la posizione dell'elemento all'interno della matrice.

MATRICI A PIU' DIMENSIONI

Possiamo definire una matrice a due dimensioni per memorizzare i valori di una tabella. Supponiamo di avere la seguente tabella:

NOME	CODICE	NAZIONE	SESSO
Anna	21SAA	Gran Bretagna	F
Giovanni	35ECK	USA	M
Riccardo	7ØWST	Svezia	M

Tabella 4 - 7

Questa tabella comprende 3 righe e 4 colonne per un totale di 12 valori stringa.

Se definiamo una matrice stringa A$, possiamo memorizzare tutti i valori della tabella come elementi di questa matrice e possiamo accedere a ciascun elemento tramite il valore dei suoi due indici.

INDICE	Ø	1	2	3
Ø	Anna	21SAA	Gran Bretagna	F
1	Giovanni	35ECR	USA	M
2	Riccardo	7ØWST	Svezia	M

Tabella 4 - 8 Matrice A$

Ogni elemento della matrice A$ è individuato da due indici, racchiusi tra parentesi dopo il nome della matrice e separati da una virgola.

Il primo indice rappresenta la riga e il secondo la colonna. Per esempio il valore di A$(Ø,1) è la stringa 21SAA, e il valore di A$(2,3) è il carattere M.

E' anche possibile definire matrici con più di due dimensioni, ma queste sono usate molto di rado.

DIM (PROGRAMMA/IMMEDIATO)

Specifica il nome di una matrice, il numero delle sue dimensioni e il valore massimo dell'indice per ogni dimensione. L'istruzione DIM può specificare una o più matrici.

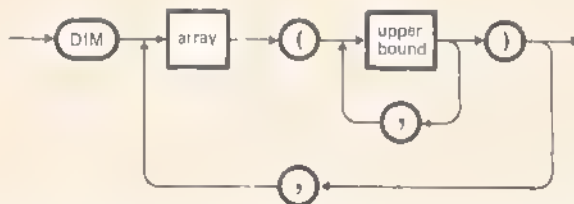



Figura 4-2 Istruzione DIM

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO	VALORI DI DEFAULT
array	è il nome di una matrice (qualsiasi nome di variabile è consentito)	
upper bound	è una qualsiasi costante numerica positiva o una variabile numerica con valore positivo. Se questo valore non è intero viene arrotondato all'intero più vicino.	se non è presente alcuna istruzione DIM, il valore massimo dell'indice per ogni dimensione viene assunto pari a 1Ø, e il numero di dimensioni viene stabilito non appena si incontra un elemento della matrice nel programma.

Esempio

SE l'utente imposta...	ALLORA...
1Ø DIM A(5), B$(2Ø,3Ø) **CR**	definisce una matrice A a una dimensione con l'indice compreso tra Ø e 5, e una matrice stringa a due dimensioni B$ con l'indice di riga compreso tra Ø e 2Ø e l'indice di colonna tra Ø e 3Ø. Nota: A è una matrice numerica a meno che un'istruzione DEFSTR stabilisca altrimenti.

Numero di Dimensioni

In BASIC è possibile definire matrici con numero arbitrario di dimensioni (compatibilmente con la memoria disponibile), ma le matrici di uso più frequente hanno soltanto una o due dimensioni.

Se l'utente non introduce alcuna istruzione DIM nel programma, la matrice

viene creata quando il BASIC incontra per la prima volta un suo elemento. In base al numero degli indici il BASIC può determinare il numero delle dimensioni. Per esempio, se in una istruzione compare:

AR1(3,5,1Ø)

allora il BASIC crea la matrice AR1 con tre dimensioni, e con un valore massimo dell'indice per ogni dimensione pari a 1Ø.

Numero di Elementi per Dimensione

SE...	E SE...	ALLORA...
nessuna istruzione DIM è presente	viene specificato OPTION BASE Ø	si hanno 11 elementi per ogni dimensione (indici tra Ø e 1Ø)
	viene specificato OPTION BASE 1	si hanno 1Ø elementi per ogni dimensione (indici tra 1 e 1Ø)
è presente una istruzione DIM	viene specificato OPTION BASE Ø	il numero di elementi per ogni dimensione è pari al valore massimo dell'indice + 1
	viene specificato OPTION BASE 1	il numero di elementi per ogni dimensione è pari al valore dell'indice.

Definizione di una Matrice

L'UTENTE DEVE	E...	OPPURE...
stabilire il valore minimo degli indici	può utilizzare l'istruzione OPTION BASE 1	può adottare il valore di default OPTION BASE Ø
assegnare un nome alla matrice stabilire il numero di dimensioni stabilire il valore massimo dell'indice per ogni dimensione	può utilizzare l'istruzione DIM	può far riferimento a un elemento della matrice nel programma. Nota: In questo caso il valore massimo dell'indice per ogni dimensione risulta pari a 1Ø.

Note

- Una istruzione DIM inizializza a zero tutti gli elementi delle matrici specificate.
- Una istruzione DIM deve precedere ogni riferimento ad elementi di matrice.
- Una istruzione DIM non può stabilire il valore massimo dell'indice per dimensione, se il controllo dell'esecuzione salta le DIM.

Esempio

VIDEO	COMMENTI
LIST 1Ø I=1 2Ø GOTO 4Ø 3Ø DIM A(5Ø) 4Ø A(1Ø)=3 50 A(11)=45 Ok RUN Subscript out of range in 5Ø Ok	L'M20 visualizzerà il messaggio d'errore: Subscript out of range in 5Ø quando viene eseguita l'istruzione 5Ø, dato che l'istruzione 3Ø è stata saltata e il valore massimo dell'indice viene assunto pari a 1Ø.

Libera lo spazio riservato a una o più matrici e rende disponibili i relativi nomi per nominare altre variabili del programma.

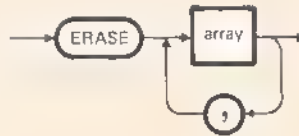



Figura 4-3 Istruzione ERASE

Esempio

VIDEO	COMMENTI
1Ø DIM A(15,15),B(1Ø,2Ø) . . . 1ØØ ERASE A,B 11Ø DIM A(1ØØ),B(2,2,2)	dopo aver eseguito l'istruzione 1ØØ, lo spazio allocato per le matrici A e B viene rilasciato. L'utente può allora definire altre variabili (in particolare matrici) con quegli stessi nomi: qui l'istruzione 11Ø definisce altre due matrici A e B con un diverso numero di dimensioni e con un diverso valore massimo degli indici.

Note

Di norma non è consigliabile riutilizzare un identificatore per altri scopi perchè può essere fonte di errori o può rendere il programma poco leggibile.

Tuttavia il ridimensionamento di matrici può risultare molto utile, ad esempio quando il nome di una matrice viene utilizzato da una subroutine e si vogliono passare a questa subroutine matrici con un diverso numero di dimensioni (o un diverso valore massimo degli indici).

OPTION BASE (PROGRAMMA/IMMEDIATO)

Dichiara il valore minimo degli indici delle matrici.

Figura 4-4 Istruzione OPTION BASE

Valori di Default

Se l'utente non scrive una istruzione OPTION BASE nel proprio programma, per default viene assunto OPTION BASE Ø; pertanto il valore minimo degli indici delle matrici è per default Ø.

Esempio

SE l'utente imposta...	ALLORA...
1Ø OPTION BASE 1 **CR**	il valore minimo degli indici è 1.
OPPURE	
OPTION BASE 1 **CR** (in modo immediato)	

Note

L'istruzione OPTION BASE 1 può essere utile soprattutto per convertire verso l'M20 programmi BASIC scritti per essere eseguiti su altri sistemi. Alcune versioni del linguaggio BASIC infatti accettano come valore minimo degli indici soltanto il valore 1.

L'istruzione OPTION BASE non può essere preceduta da un'istruzione DIM o da un riferimento a un elemento di matrice.

5. COME VENGONO INTRODOTTI I DATI IN BASIC

SOMMARIO

Questo capitolo si propone di descrivere alcune modalità con le quali vengono forniti in BASIC i dati al computer.

Verranno prese in esame:

- le istruzioni CLEAR, LET e SWAP

- le istruzioni INPUT e LINE INPUT

- le istruzioni DATA, READ e RESTORE

Altre modalità di introduzione dei dati (che comportano l'utilizzazione dei file esterni) saranno esaminate successivamente (vedere il Capitolo 12).

INDICE

ISTRUZIONI DI ASSEGNAZIONE

In BASIC vi sono tre istruzioni di assegnazione:

- l'istruzione CLEAR che consente di azzerare le variabili numeriche e di inizializzare le variabili stringa al valore stringa nulla.

- l'istruzione LET, che consente di assegnare il valore di una espressione ad una variabile. La variabile e l'espressione devono essere dello stesso tipo (entrambe numeriche o entrambe stringa).

- l'istruzione SWAP, che consente di scambiare i valori di due variabili, purchè siano dello stesso tipo (intera, in semplice precisione, in doppia precisione, stringa).

Le istruzioni LET e SWAP vengono spesso utilizzate per eseguire calcoli immediati.

CLEAR (PROGRAMMA/IMMEDIATO)

Azzera tutte le variabili numeriche, inizializza le variabili stringa al valore stringa nulla, chiude tutti i file dati e tutte le finestre (v. capitolo 14) e azzera il video. E' anche possibile, con l'istruzione CLEAR, stabilire lo spazio di memoria dedicato al BASIC e quello dedicato allo stack.

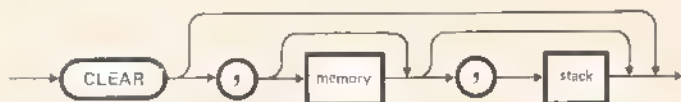



Figura 5-1 Istruzione CLEAR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
memory	rappresenta l'ammontare di memoria (in byte) disponibile per i programmi BASIC. La memoria, utilizzabile dal BASIC, può anche essere stabilita con il comando SBASIC del PCOS. Se il parametro memory viene omesso, il suo valore è quello stabilito con il comando SBASIC oppure 36800 byte (come seconda alternativa)
stack	stabilisce lo spazio, dedicato allo stack. Il valore di default è il valore minore tra 512 byte e 1/8 della memoria disponibile. Lo stack è una parte della memoria dedicata al BASIC, utilizzata per memorizzare l'indirizzo di ritorno dei sottoprogrammi, delle funzioni ecc.

Esempi

VIDEO	COMMENTI
CLEAR	azzera le variabili numeriche, inizializza le variabili stringa al valore stringa nulla, chiude i file dati e le finestre e azzera il video. La memoria può essere stata dimensionata in modo esplicito tramite il comando SBASIC oppure viene assunta per default pari a 36800 byte. Il valore dello stack viene assunto per default.
CLEAR ,32768	come sopra, ma la memoria viene dimensionata a 32768 byte.

CLEAR ,,2000	come nel primo esempio, ma lo stack è stato dimensionato a 2000 byte.
CLEAR ,32768,2000	come nel primo esempio, ma la memoria viene dimensionata a 32768 byte e lo stack a 2000 byte.

Note

All'inizio dell'esecuzione di un programma BASIC, tutte le variabili numeriche vengono azzerate e tutte le variabili stringa vengono inizializzate al valore stringa nulla (ad eccezione delle variabili definite nell'area COMMON in fase di CHAINing di un programma ad un altro, vedere il Capitolo 11). Non è quindi necessario usare un'istruzione CLEAR all'inizio del programma al solo scopo di azzerare le variabili numeriche e di inizializzare le variabili stringa al valore stringa nulla.

Il BASIC alloca in modo dinamico lo spazio riservato alle stringhe. Viene visualizzata una segnalazione d'errore (Out of string space) nel caso in cui non ci sia memoria sufficiente per il BASIC.

LET (PROGRAMMA/IMMEDIATO)

Assegna un valore ad una variabile.

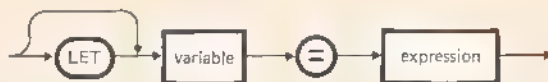



Figura 5-2 Istruzione LET

Esempi

SE l'utente imposta...	ALLORA...
LET K = 1.5 CR	il valore 1.5 è assegnato alla variabile numerica K
LET X = K + 2 CR	il valore dell'espressione numerica K + 2 è assegnato alla variabile numerica X
A$(1) = "ABC" CR	il valore della costante stringa "ABC" è assegnato alla variabile stringa con indice A$(I) Nota: La parola chiave LET è opzionale

Assegnazioni numeriche

Se il valore dell'espressione numerica non è dello stesso tipo della variabile di destinazione (cioè la variabile che compare alla sinistra del segno di uguaglianza) il BASIC converte il tipo del valore dell'espressione in modo da renderlo uguale a quello della variabile di destinazione in base alle regole precedentemente descritte (vedere il paragrafo CONVERSIONI NUMERICHE del Capitolo 4). Se la variabile di destinazione non può contenere il valore calcolato, può verificarsi un arrotondamento oppure un overflow.

Assegnazioni stringa

L'assegnazione stringa viene effettuata trasferendo il valore dell'espressione stringa nella variabile di destinazione, carattere per carattere da sinistra a destra. L'operazione termina quando tutti i caratteri sono stati trasferiti.

Note

Non sono ammesse assegnazioni simultanee. Se ad esempio si impostasse:

```
100 LET B% = C% = 0 CR
```

il BASIC INTERPRETEREBBE IL SECONDO SEGNO DI UGUAGLIANZA COME UN OPERATORE DI CONFRONTO e assegnerebbe a B% il valore -1 (cioè vero) se C% è uguale a Ø, ed il valore Ø (cioè falso) se C% è diverso da zero. (Per ulteriori dettagli sulle espressioni di confronto vedere il Capitolo 6).

SWAP (PROGRAMMA/IMMEDIATO)

Permette di scambiare i valori di due variabili semplici. L'istruzione SWAP può essere eseguita con qualsiasi tipo di variabile (intera, in semplice precisione, in doppia precisione, stringa), ma le due variabili devono essere dello stesso tipo, altrimenti verrà visualizzato l'errore "Type mismatch" (incompatibilità di tipo). Le variabili devono anche essere inizializzate, altrimenti verrà visualizzato l'errore "Illegal function call".

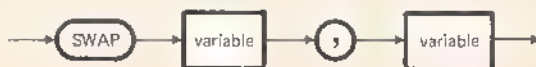



Figura 5-3 Istruzione SWAP

Esempio

VIDEO	COMMENTI
LIST 1Ø A$ = " ONE " 2Ø B$ = " ALL" 3Ø C$ = " FOR " 4Ø PRINT A$;C$;B$ 5Ø SWAP A$;B$ 6Ø PRINT A$;C$;B$ Ok RUN ONE FOR ALL ALL FOR ONE Ok	L'istruzione 5Ø esegue l'operazione di SWAP fra le variabili A$ e B$. L'istruzione 4Ø visualizza ONE FOR ALL, l'istruzione 6Ø visualizza ALL FOR ONE.

IL FILE DATI INTERNO

Molti problemi rendono necessaria l'introduzione di una grande quantità di dati nel computer. Per fare ciò, l'utente può utilizzare ripetutamente le istruzione LET, INPUT e LINE INPUT.

E' evidente, però, che tale procedura può risultare, in certi casi, molto pesante. Un metodo molto più conveniente ed efficace per introdurre dati è quello di servirsi delle istruzioni DATA, READ e RESTORE.

Le istruzioni DATA creano un file "interno", cioè una sequenza di dati (del programma), che devono essere trasferiti nelle variabili di programma con l'uso di una o più istruzioni READ.

L'istruzione RESTORE riposiziona il puntatore ("pointer") all'inizio del file o al numero di linea specificato.

DATA/READ/RESTORE (PROGRAMMA)

DATA crea un file dati interno.

READ legge i dati dal file interno (creato con una o più istruzioni DATA) e li assegna alle variabili specificate.

RESTORE sposta il pointer all'inizio di un file dati interno o al numero di linea specificato.

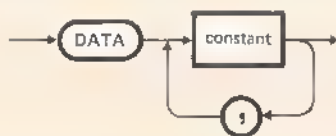



Figura 5-4 Istruzione DATA

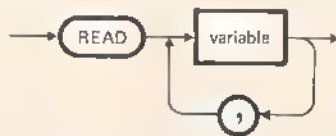



Figura 5-5 Istruzione READ

Figura 5-6 Istruzione RESTORE

Esempi

VIDEO	COMMENTI
LIST 10 READ A,B,C,D,E,F,G,H,I,J 20 DATA 1,2,3,4,5,6,7,8,9,10 30 PRINT A;B;C;D;E;F;G;H;I;J Ok RUN 1 2 3 4 5 6 7 8 9 10 Ok	i valori da 1 a 10 vengono assegnati a dieci variabili
LIST 10 DATA 1,2,3,4 20 READ A,B,C,D,E,F,G,H,I,J 30 DATA 5,6,7 40 DATA 8,9,10 50 PRINT A;B;C;D;E;F;G;H;I;J Ok RUN 1 2 3 4 5 6 7 8 9 10 Ok	le istruzioni 10, 20, 30 e 40 hanno lo stesso effetto delle istruzioni 10 e 20 dell'esempio precedente. Nota: Un'istruzione DATA in un programma non deve necessariamente corrispondere ad una ben determinata istruzione READ. Ciò avviene perchè prima dell'esecuzione di un programma viene creato un file dati (il file dati "interno"). Questo file contiene i valori relativi a tutte le istruzioni DATA del programma secondo la sequenza dei numeri di linea. Nel corso dell'esecuzione del programma, l'istruzione READ acquisisce i suoi valori da questo file.

```
LIST
10 READ A,B,C
20 DATA 1,2,3,4,5,6,7,8,9,10
30 PRINT A;B;C
40 READ D,E,F,G
50 PRINT D;E;F;G
RUN
 1  2  3
 4  5  6  7
Ok
```

l'istruzione 10 assegna i valori 1,2,3 ad A,B e C, l'istruzione 40 assegna i valori 4,5,6 e 7 a D,E,F,G rispettivamente.

Nota: Non è necessario leggere, uno alla volta, i valori del file dati.

```
LIST
10 READ A,B,C,D,E
20 DATA 1,2,3,4
Ok
RUN
Out of data
Ok
```

l'M20 visualizza un messaggio di errore:

Out of data

e ritorna in Stato Comandi, perchè ci sono più variabili che dati.

```
LIST
10 READ A,B,C
20 DATA 15,25,35,5,6,12
30 PRINT A;B;C
40 RESTORE
50 READ X,Y,Z
60 PRINT X;Y;Z
Ok
RUN
 15  25  35
 15  25  35
Ok
```

l'istruzione 10 fa si che alla variabile A venga assegnato il valore 15, a B il valore 25 e a C il valore 35. L'istruzione RESTORE della linea 40 fa si che i valori vengano assegnati partendo nuovamente dall'inizio del file. Quindi l'istruzione 60 fa si che gli stessi valori assegnati ad A,B,C (15,25,35) siano assegnati rispettivamente a X,Y,Z.

Se non si fosse utilizzata l'istruzione RESTORE, ad X sarebbe stato assegnato il valore 5, ad Y il valore 6 e a Z il valore 12.

```
LIST
10 READ X1$,Y1$,Z1
20 DATA "DENVER,", COLORADO, 80211
30 PRINT X1$;Y1$;Z1
Ok
RUN
DENVER,COLORADO 80211
Ok
```

l'istruzione 10 fa si che ad X1$ sia assegnato il valore DENVER, (inclusa la virgola finale), ad Y1$ il valore COLORADO e a Z1 il valore 80211.

Nota: Le istruzioni READ possono contenere sia variabili numeriche che variabili stringa.

Le istruzioni DATA possono contenere sia dati numerici che dati stringa.

Il tipo di un dato in ingresso (nella sequenza dei dati) deve essere coerente al tipo di variabile alla quale deve essere assegnato; cioè le variabili numeriche richiedono, come dati, costanti numeriche (è consentita la conversione da un tipo numerico ad un altro, per esempio é possibile avere una costante in singola precisione associata a una variabile intera), mentre le variabili stringa richiedono dati stringa tra virgolette o senza virgolette. Una stringa deve essere tra virgolette se contiene virgole (ad esempio DENVER,) oppure spazi iniziali o finali (ad esempio lo spazio che precede COLORADO nell'istruzione 2Ø non viene assegnato alla variabile Y1$ perchè COLORADO è una stringa non racchiusa tra virgolette).

ISTRUZIONI DI INPUT

L'istruzione DATA utilizza costanti per assegnare valori a variabili. Quando si introduce un programma, occorre conoscere quali valori si intendono assegnare. Inoltre i valori contenuti nel file dati interno, sono registrati su disco insieme con il programma. Questi valori sono quindi permanenti e possono essere modificati solo cambiando una o più istruzioni DATA del programma.

Le istruzioni INPUT e LINE INPUT offrono una maggior flessibilità in quanto consentono di introdurre valori solo al momento dell'esecuzione del programma.

Quando il controllo dell'esecuzione arriva su una di queste istruzioni, l'esecuzione del programma viene sospesa e l'M20 rimane in attesa di dati da tastiera. Dopo che il programma è stato registrato in memoria, è possibile eseguirlo ogni volta che lo si desideri ed è possibile fornire i valori al computer sul momento senza dover modificare il programma. Questa flessibilità consente, pertanto, di scrivere un programma generale atto a risolvere un problema particolare ancora prima di conoscere i valori specifici che il programma dovrà utilizzare. Se si devono, però, introdurre molti dati è preferibile utilizzare un file dati interno (dati permanenti) o uno o più file dati esterni (vedere il Capitolo 12).

L'istruzione INPUT consente l'introduzione di uno o più dati numerici o dati stringa (separati da una virgola) che saranno assegnati alla variabile o alle variabili indicate nell'istruzione.

L'istruzione LINE INPUT permette di introdurre un'intera linea di input e di assegnarla ad una variabile stringa. E' possibile inserire un messaggio atto a ricordare il significato dei dati da impostare (messaggio prompt) sia nell'istruzione INPUT che nella LINE INPUT; questo messaggio appare sul video quando l'istruzione viene eseguita.

INPUT (PROGRAMMA)

Legge dati da tastiera e li assegna ad una o a più variabili specificate.

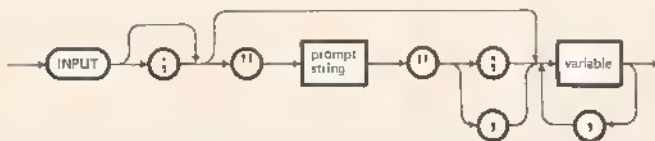



Figura 5-7 Istruzione INPUT

Un Punto Interrogativo

Un punto interrogativo (seguito da uno spazio) viene automaticamente visualizzato come uno "standard prompt" in fase di esecuzione di un'istruzione INPUT, anche se l'istruzione non include la clausola "prompt string".

VIDEO	COMMENTI
LIST 10 INPUT X 20 PRINT X "SQUARED IS" X^2 30 END Ok RUN ? 5 5 SQUARED IS 25 Ok	quando si esegue l'istruzione 10 lo "standard prompt" (?) viene visualizzato e con ciò si indica che il programma è in attesa di un dato. In questo caso non si utilizza la clausola "prompt string" nell'istruzione INPUT (vedere l'istruzione 10).

Self Prompting

Inserendo una "prompt string" in una istruzione INPUT, l'utente può visualizzare il messaggio che desidera (self prompting) per ricordare il valore (o i valori) da introdurre.

VIDEO	COMMENTI
LIST 10 PI = 3.1415 20 INPUT "Radius";R 30 A = PI*R^2 40 PRINT "Area";A 50 GOTO 20 Ok RUN Radius? 7.4 Area 172.029 Radius? ecc.	l'esecuzione dell'istruzione 20 fa visualizzare il messaggio Radius davanti allo standard prompt (?)

Eliminazione dello Standard Prompt

L'utente può eliminare lo standard prompt (?), scrivendo una virgola (anzichè un punto e virgola) dopo il prompt.

VIDEO	COMMENTI
LIST 1Ø INPUT "Date ", D$ 2Ø PRINT D$ Ok RUN Date 3Ø/Oct/69 3Ø/Oct/69 Ok	lo standard prompt (?) viene eliminato perchè nell'istruzione 1Ø la stringa "Date" (prompt string) è seguita da una virgola (,) anzichè da un punto e virgola (;).

Eliminazione dell'Eco di CR

L'utente può eliminare l'eco di CR su video introducendo un punto e virgola (;) dopo la parola chiave INPUT.

VIDEO	COMMENTI
LIST 1Ø INPUT; "Date";D$ 2Ø PRINT " J.C." Ok RUN Date? 3Ø/Oct/69 J.C.	l'eco su video del ritorno a capo/interlinea viene eliminato inserendo un punto e virgola (;) immediamente dopo la parola chiave INPUT (vedere l'istruzione 1Ø). La successiva operazione PRINT/INPUT (vedere l'istruzione 2Ø) sarà eseguita dalla successiva posizione su video.

Introduzione di una Lista di Dati

Una istruzione INPUT consente l'inserimento da tastiera di uno o più dati numerici o dati stringa.

VIDEO	COMMENTI
LIST 1Ø INPUT A,B$,C(3) 2Ø PRINT A;B$;C(3) 3Ø GOTO 1Ø Ok RUN ? 1.2,ABC,4 1.2 ABC 4 ? ABD,1.3,5 ?Redo from start ? 1.3,ABD,5 1.3 ABD 5 ? ^C Break in 1Ø Ok	quando si esegue l'istruzione 1Ø l'utente deve introdurre tre dati. Il primo deve essere numerico (1.2), il secondo di tipo stringa (ABC) (non è necessario che sia tra virgolette), il terzo numerico (4). Verranno assegnati rispettivamente alle variabili A,B$ e C(3). Quando si esegue per la seconda volta l'istruzione 1Ø supponiamo di assegnare un dato non coerente (ABD), e cioè una stringa al posto di un numero. Il sistema visualizza: ? Redo from start e l'utente deve introdurre nuovamente i dati. Per interrompere l'esecuzione del programma occorre premere CTRL C Per riprenderne l'esecuzione premere C O N T CR Nota: Il tipo di dato, che viene introdotto da tastiera, deve essere dello stesso tipo della variabile di destinazione, cioè le variabili numeriche richiedono, come dati, costanti numeriche (è possibile la conversione da un tipo numerico ad un altro, ad esempio una costante in doppia

precisione può essere associata ad una variabile intera), e le variabili stringa richiedono, come dati, stringhe racchiuse o no tra virgolette. E' necessario che la stringa sia racchiusa tra virgolette se contiene virgole o spazi iniziali o finali.

Valori numerici possono essere introdotti in variabili stringa tramite un'istruzione INPUT. Se introduciamo un numero in una variabile stringa, questo viene interpretato come la sequenza dei corrispondenti caratteri ASCII. Per riottenere il valore numerico, l'utente dovrà usare la funzione VAL (vedere il Capitolo 9), altrimenti si avrebbero errori di incompatibilità di tipo.

? Redo from Start

Se ad un richiesta di INPUT si risponde con troppi o con pochi dati, oppure con un dato di tipo non corretto (un dato stringa al posto di un dato numerico), su video verrà visualizzato il messaggio d'errore "? Redo from Start". Fin quando l'utente non fornirà una risposta corretta, non verrà assegnato alcun valore alle variabili.

LINE INPUT (PROGRAMMA)

Permette l'introduzione di una intera linea, fino al ritorno a capo/interlinea e la assegna ad una variabile stringa senza far uso di delimitatori (la lunghezza massima di una linea è di 255 caratteri).

Figura 5-8 Istruzione LINE INPUT

Un Punto Interrogativo all'interno di un Prompt

Lo standard prompt (?) non compare quando si esegue una istruzione LINE INPUT. Se si desidera visualizzare il punto interrogativo si dovrà includerlo alla fine della clausola "prompt string".

VIDEO	COMMENTI
LIST 1Ø LINE INPUT "Name? ";N$ 2Ø PRINT "JONES" Ok RUN Name? LINDA JONES Ok	la stringa prompt (Name?) viene visualizzata prima dell'introduzione dei dati. Tutti i caratteri introdotti dalla fine del prompt al CR vengono assegnati alla variabile stringa (N$).

Eliminazione dell'Eco di CR

E' possibile eliminare l'eco di CR su video introducendo un punto e virgola (;) subito dopo LINE INPUT.

VIDEO	COMMENTI
LIST 1Ø LINE INPUT;"Name? ";N$ 2Ø PRINT " JONES" Ok RUN Name? LINDA JONES Ok	l'eco sul video del ritorno a capo/interlinea viene eliminato inserendo un punto e virgola (;) subito dopo LINE INPUT (vedere l'istruzione 1Ø). La prossima operazione PRINT/INPUT (vedere l'istruzione 2Ø) incomincerà a visualizzare i caratteri dalla posizione successiva su video.

6. ESPRESSIONI

SOMMARIO

Questo capitolo classifica le espressioni usate in BASIC in numeriche, stringa, di confronto o logiche.

Vengono descritte le regole che l'utente deve rispettare nella formulazione dell'espressione ed il livello di priorità che il BASIC assume nel loro svolgimento.

INDICE

ESPRESSIONI NUMERICHE

La maggior parte dei programmi richiede una qualche forma di calcolo numerico.

Come si è potuto notare negli esempi fatti, alla sinistra del segno di uguaglianza di un'istruzione LET appaiono solo variabili.

Alla destra del segno uguale possono invece apparire sia variabili che costanti. Infatti, esse possono essere collegate da simboli speciali chiamati operatori, che indicano quale operazione numerica deve essere svolta.

Si considerino i seguenti esempi:

```
 7Ø LET L=ACCOUNT
 6Ø LET Y = 16+1.7+12
2ØØ M = 83-44+37/N
 2Ø LET X = X+1
```

L'ultima istruzione è di particolare interesse. La maggior parte delle istruzioni LET ha le caratteristiche delle equazioni algebriche, quest'ultima no. L'equazione X = X+1 è priva di significato da un punto di vista algebrico.

L'istruzione LET assegna un valore ad una variabile: non implica che i valori su entrambi i lati del segno di uguaglianza siano matematicamente uguali. Quest'ultima istruzione è valida e significativa e può essere interpretata nel seguente modo: si aggiunga 1 al valore della variabile X e si assegni ad X questo nuovo valore. Il nuovo valore di X sostituirà quello vecchio. Il segno di uguale è un operatore.

La parte dell'istruzione LET alla destra del segno di uguale si chiama espressione. L'espressione specifica quale valore deve essere assegnato alla variabile che si trova alla sua sinistra. (Lo svolgimento di una espressione porta ad un solo valore numerico). Una espressione numerica può essere composta da un solo numero o da una sola variabile numerica o da una combinazione di numeri e di variabili numeriche e operatori. Si deve però tenere presente che occorre assegnare un valore ad una variabile numerica prima che questa possa essere usata in una espressione. In caso contrario, alla variabile numerica viene automaticamente attribuito il valore Ø.

Alcuni esempi di espressioni numeriche sono presentati qui di seguito:

X
X+Y+SPEED
6.4^2
-7+A/CYAR

Operatori Numerici

Come si è notato in precedenza, il BASIC si serve di operatori per definire le operazioni numeriche. Vi sono otto operazioni numeriche, ognuna caratterizzata da un proprio simbolo.

SIMBOLO	OPERAZIONE	ESEMPI
+	addizione	X = 3.2 Ok ?X+1.1 4.3 Ok
-	sottrazione	?X-1.3 1.9 Ok
\	divisione tra interi Gli operandi sono arrotondati al numero intero più vicino (che deve essere compreso nell'intervallo da -32768 a 32767) prima di effettuare la divisione e anche il quoziente viene troncato al valore intero più vicino.	?1Ø\4 2 Ok ? 25.68\6.99 3 Ok
MOD	divisione modulo Fornisce quel numero intero che è il resto di una divisione tra numeri interi	?1Ø.4 MOD 4 2 Ok (10/4=2 con resto 2) ?25.68 MOD 6.99 5 Ok (26/7=3 con resto 5)

*	moltiplicazione	?X*3.92 12.544 Ok
/	divisione	?3/6.Ø5 Ø.495868 Ok
-	cambiamento di segno	?-X -3.2 Ok
^	elevamento a potenza	?X^3 32.768 Ok

Note

Accertatevi di usare il simbolo * quando si vuole eseguire una moltiplicazione. In matematica, 6X è una espressione valida; nel linguaggio BASIC, si deve, invece, usare 6*X per esprimere la moltiplicazione per sei volte della variabile X.

Per comodità, tutti gli operatori numerici usati in BASIC sono disponibili sia nella sezione numerica della tastiera dell'M20 che in quella alfanumerica (ad eccezione del simbolo di elevamento a potenza, che compare solo nella sezione alfanumerica, e di MOD che deve essere introdotto digitando i tre caratteri che lo compongono).

Livello di Priorità degli Operatori Numerici

Quando nell'ambito di una espressione vengono utilizzati due o più operatori, spesso si potrebbero verificare situazioni ambigue. Ad esempio, l'espressione:

3*L - 6*W

deve intendersi

(3*L) - (6*W)

oppure

3*(L-6*W)?

Il linguaggio BASIC ha, però, dei livelli di priorità, che vengono seguiti nell'esecuzione delle diverse operazioni numeriche. Le operazioni numeriche e i livelli di priorità vengono riportati qui di seguito (nell'ordine di priorità decrescente).

LIVELLO DI PRIORITA'	OPERAZIONE	COMMENTI
IL PIU' ALTO	elevamento a potenza	
	cambiamento di segno	
	moltiplicazione e divisione	la moltiplicazione e la divisione hanno lo stesso livello di priorità
	divisione tra interi	
	divisione modulo	
IL PIU' BASSO	addizione e sottrazione	l'addizione e la sottrazione hanno lo stesso livello di priorità

Note

Riferendoci all'esempio precedente, possiamo ora dire che l'espressione 3*L - 6*W deve intendersi (3*L) - (6*W).

Nel caso di operatori con la stessa priorità (ad. es. / e *) le operazioni vengono eseguite nell'ordine da sinistra a destra. Quindi, 9/3*3 è l'equivalente di (9/3)*3: entrambe portano allo stesso risultato di 9.

Uso delle Parentesi per Modificare il Livello di Priorità

Si possono presentare dei casi in cui si vuole cambiare il livello normale di priorità con cui vengono eseguite le operazioni. Per ottenere questo, si devono usare le parentesi con le stesse modalità dell'algebra. In questo caso, il calcolo inizia con lo svolgimento delle operazioni racchiuse tra le parentesi più interne per poi passare via

via a quelle più esterne. All'interno di una coppia di parentesi, viene rispettato il livello normale di priorità delle operazioni.

Qui di seguito viene presentato un semplice esempio di utilizzazione delle parentesi.

Supponiamo di voler calcolare $(5X)^2$. Se in BASIC questa espressione viene impostata sotto forma di 5*X^2, la X viene innanzi tutto elevata al quadrato, ed il risultato viene successivamente moltiplicato per 5, dal momento che l'elevamento a potenza ha un livello di priorità superiore alla moltiplicazione. Per modificare questa situazione, è sufficiente impostare l'espressione sotto forma di (5*X)^2. In questo caso, la X viene dapprima moltiplicata per 5 ed il risultato viene poi elevato al quadrato.

E' evidente che l'equivalente in BASIC risulterà tanto più complesso quanto più complessa sarà l'espressione numerica. Negli esempi riportati qui di seguito, si evidenziano espressioni numeriche ed i loro equivalenti in BASIC. Tali esempi dovrebbero contribuire ad una più agevole comprensione delle regole di priorità.

Esempi

ESPRESSIONE NUMERICA	EQUIVALENTE IN BASIC	INTERPRETAZIONE
$\frac{x + y + z}{2}$	(X+Y+Z)/2	1. Sommare X,Y e Z 2. Dividere la somma per 2
$x + \frac{y + z}{2}$	X+(Y+Z)/2	1. Sommare Y e Z 2. Dividere la somma per 2 3. Sommare X al risultato
$2x + 5$	2*X+5	1. Moltiplicare X per 2 2. Sommare 5 al risultato
$2(x + 4)$	2*(X+4)	1. Sommare 4 e X 2. Moltiplicare la somma per 2
$x^2 + 3$	X^2+3	1. Elevare X al quadrato 2. Sommare 3 al risultato

$(x + 3)^2$	(X+3)^2	1. Sommare 3 e X 2. Elevare al quadrato il risultato
$\frac{(x + 3)^2}{4}$	(X+3)^2/4	1. Sommare 3 e X 2. Elevare la somma al quadrato 3. Dividere il risultato per 4
$\frac{x^2}{6} \cdot \frac{x + y}{2}$	(X^2/6)*((X+Y)/2))	1. Elevare X al quadrato e dividere il risultato per 6 2. Sommare X e Y e dividere per 2 3. Moltiplicare il primo risultato per il secondo

Note

E' buona norma di programmazione fare uso delle parentesi ogni volta che vi sono dubbi sullo svolgimento dell'espressione, anche quando esse non sono strettamente necessarie.

Le espressioni usate in un programma, possono essere molto complesse. Nell'eventualità che un programma venga registrato in memoria e sia utilizzato solo sporadicamente, è facile dimenticare i calcoli che esso esegue. Per questo motivo, quando si scrive un programma è utile usare l'istruzione REM del linguaggio BASIC ed i campi commento.

Tipo di Espressione

Il tipo di un'espressione numerica, e cioè il tipo del risultato ottenuto dallo svolgimento di un'espressione (prima della sua assegnazione ad una variabile) dipende dal tipo degli operandi.

Si possono ipotizzare quattro situazioni a seconda del tipo dei due operandi in questione.

L'espressione, nell'eventualità in cui in essa intervengano più di due operandi, può essere considerata come una serie di calcoli in cui intervengono due operandi.

La tabella che segue fornisce un sommario delle quattro situazioni possibili.

SE...	ALLORA...	VIDEO
entrambi gli operandi sono dello stesso tipo numerico (intero, in precisione semplice, in precisione doppia)	il risultato è dello stesso tipo degli operandi	A # = 3.29745219 Ok B # = 4.57297190-1 Ok ?A#+B# 3.75474938 Ok
un operando è intero mentre l'altro è in semplice precisione	il risultato è in semplice precisione	I% = 25 Ok C! = 4.2975 Ok ?I%-C! 2Ø.7Ø25 Ok
un operando è intero mentre l'altro è in doppia precisione	il risultato è in doppia precisione	?I%*A# 82.4363Ø475 Ok
un operando è in semplice precisione e l'altro in doppia precisione	il risultato è in doppia precisione	?C!/B# 9.3976Ø887993736 Ok

Arrotondamento, Overflow e Underflow

I numeri in virgola mobile costituiscono una forma di approssimazione dei numeri reali della matematica.

SE...	ALLORA...
in un'espressione numerica, uno o più operandi sono in virgola mobile	il calcolo è approssimato e si può perdere in precisione. In questo caso non si tiene conto delle cifre meno significative e l'ultima cifra considerata viene arrotondata.

il valore dell'espressione supera il valore massimo, che può essere memorizzato nella variabile ricevente	viene visualizzato un messaggio di "Overflow"; come risultato viene fornito, con il segno algebrico corretto, l'infinito di macchina, dopo di che l'esecuzione continua
viene eseguita una divisione per zero	viene visualizzato il messaggio "Division by Zero"; come risultato della divisione viene fornito l'infinito di macchina con il segno del numeratore, dopo di che l'esecuzione continua.
il calcolo di un elevamento a potenza dà luogo ad un valore nullo elevato ad una potenza negativa	viene visualizzato il messaggio "Division by Zero"; come risultato dell'elevamento a potenza viene fornito l'infinito di macchina, con segno positivo dopodichè l'esecuzione continua.
il valore dell'espressione è minore del più piccolo valore rappresentabile	il valore diviene zero (Underflow) dopo di che l'esecuzione continua.
in un'assegnazione numerica, il tipo dell'espressione è diverso dal tipo della variabile ricevente	il tipo dell'espressione viene, automaticamente, convertito al tipo della variabile ricevente.

Nota: L'infinito di macchina viene visualizzato come 3.40282E+38.

Valori non Definiti

Qualora in una espressione numerica una variabile numerica non sia ancora stata inizializzata, essa viene inizializzata al valore zero.

Forme Indeterminate

Lo svolgimento di un'espressione numerica può sfociare in una forma indeterminata del tipo:

Ø/Ø : viene visualizzato il messaggio "Division by Zero" e viene fornito il valore 3.4Ø282E+38 (infinito di macchina).

Ø∧Ø: si assume un valore uguale ad 1.

ESPRESSIONI STRINGA

Il linguaggio BASIC consente l'utilizzazione di espressioni stringa che, per molti versi, sono analoghe alle espressioni numeriche appena esaminate. Una espressione stringa può essere o una costante stringa, o una variabile semplice stringa, o un elemento di una matrice stringa, o una funzione stringa o una concatenazione degli elementi suddetti mediante l'utilizzo del segno di addizione (+).

Utilizzando questo segno, le stringhe possono essere "concatenate". Qui di seguito, vengono mostrati alcuni esempi di espressioni stringa usate nell'ambito di un'istruzione LET.

```
 5Ø LET A$ = "Chicago,"
 9Ø B$ = "IL.,"
1ØØ N$ = A$+B$+"USA"
```

La concatenazione, indicata nell'istruzione 1ØØ, fa si che ad N$ venga assegnata la stringa

Chicago,IL.,USA

La lunghezza di una stringa risultante da una concatenazione di una o più stringhe è uguale alla somma della lunghezza delle singole stringhe. Lo svolgimento dell'espressione procede da sinistra verso destra.

Si deve evitare di assegnare più di 255 caratteri ad una variabile stringa, altrimenti il sistema evidenzierà il messaggio d'errore:

String too long

Nota

L'operando di una stringa che compare in una espressione stringa può essere la stringa nulla (""). La stringa nulla può anche essere il valore di default di una variabile stringa non inizializzata.

ESPRESSIONI DI CONFRONTO

Una espressione di confronto paragona due espressioni numeriche o stringa mediante l'uso di operatori di confronto.

Operatori di Confronto

Gli operatori di confronto, elencati qui di seguito, sono:

=	uguale (il segno di uguale viene anche utilizzato per assegnare un valore ad una variabile, vedere l'istruzione LET)
>	maggiore di
<	minore di
>= (oppure =>)	maggiore di o uguale a
<= (oppure =<)	minore di o uguale a
<> (oppure ><)	non uguale a.

Non è consentito confrontare una espressione numerica con una espressione stringa e viceversa. Ad esempio:

A + B>C	è valido
C + D>= E + F	è valido
A$ = B$	è valido
B$>C1	non è valido, se C1 è una variabile numerica.

Il confronto tra numeri ha un significato ovvio. E' anche possibile il confronto di stringhe di caratteri. Esso dipenderà dal valore numerico della rappresentazione del carattere (che viene stabilita in base al valore decimale ASCII di ogni carattere nell'ambito di una stringa). Il confronto delle stringhe viene effettuato da sinistra verso destra, carattere per carattere e termina al primo carattere diverso. Il risultato del confronto è fatto in base a questi caratteri.

Innanzi tutto vengono svolte le espressioni numeriche o stringa, poi vengono applicati gli operatori di confronto ai risultati di tali espressioni.

Ad esempio, lo scrivere

A>B + C

è equivalente allo scrivere

A>(B + C)

L'espressione di confronto fornisce un risultato numerico. Esso può essere -1 (se il confronto è vero) oppure Ø (se invece è falso).

Esempi

Consideriamo alcuni esempi, che prevedono l'uso di espressioni di confronto. Assegniamo, innanzi tutto, dei valori alle variabili X ed Y.

VIDEO	COMMENTI
X=1 Ok Y=2 Ok	Il BASIC esegue le assegnazioni specificate
?X>Y Ø Ok	Il BASIC visualizza Ø (cioè falso), poichè X non è maggiore di Y
?X<>Y -1 Ok	Il BASIC visualizza -1 (cioè vero), poichè X è diverso da Y
?SIN(X)<Ø Ø Ok	Il BASIC visualizza Ø (cioè falso), poichè SIN(X) è positivo
?X MOD Y=1 -1 Ok	Il BASIC visualizza -1 (cioè vero), poichè X MOD Y è uguale ad 1
?"TOKYO">"FRANKFURT" -1 Ok	Il BASIC visualizza -1 (cioè vero), poichè TOKYO è maggiore di FRANKFURT (cioè segue FRANKFURT nell'ordine alfabetico)

?"TOKYO">"TOKY01" Ø Ok	Il BASIC visualizza Ø (cioè falso), poichè TOKYO è inferiore a TOKY01. Quando due stringhe sono di diversa lunghezza e la più corta è identica alla prima parte di quella più lunga allora quest'ultima è considerata maggiore

L'uso di Espressioni di Confronto

Il risultato di una espressione di confronto può essere utilizzato per prendere una decisione a proposito di un flusso di programma. E' possibile usare le espressioni di confronto nell'ambito delle seguenti istruzioni di controllo:

- IF...GOTO...ELSE, oppure
- IF...THEN...ELSE, oppure
- WHILE

dove viene controllata una condizione al fine di determinare le successive operazioni del programma (vedere il Capitolo 8).

La condizione può consistere in una espressione numerica, di confronto o logica. Il BASIC determina se la condizione (dopo IF oppure WHILE) è vera o falsa controllando il risultato dell'espressione per verificare se sia uguale a zero o meno. Un risultato diverso da zero è assunto come valido, mentre lo zero viene considerato falso.

Ad esempio, la seguente istruzione:

```
100 IF A$>B$ THEN 50
```

trasferisce il controllo dell'esecuzione all'istruzione 50 se la condizione (A$>B$) è vera. Se invece la condizione è falsa (e cioè A$ non è maggiore di B$) viene eseguita l'istruzione successiva.

ESPRESSIONI LOGICHE

Un'espressione logica è formata da un operando preceduto dall'operatore logico NOT, oppure da due operandi separati da un altro operatore logico (AND, OR, XOR, EQV, e IMP) o da due operandi separati da un operatore logico e da NOT.

Gli operandi in un'espressione logica possono essere espressioni numeriche o di confronto. Entrambe forniscono un risultato numerico.

Il risultato di un'espressione logica è anch'esso numerico: è un valore intero con qualsiasi combinazione di bit nell'intervallo da -32768 a 32767.

Di seguito sono presentati alcuni esempi di espressioni logiche.

NOT X	è valido
X AND Y	è valido
A>B OR C>D	è valido
I% AND A$<B$	è valido
A$ XOR B$	non è valido (poichè gli operandi sono stringhe).

Operatori Logici

Gli operatori logici convertono i loro operandi in stringhe di 16 bit, con segno, usando la rappresentazione intera complemento a due nell'intervallo da -32768 a +32767. Quando gli operandi si trovano fuori da questo intervallo si verificherà un errore.

L'operazione data viene svolta su questi numeri interi dove ogni bit del risultato è determinato dai bit corrispondenti dei due operandi.

Gli operatori logici sono elencati nella tabella qui di seguito, denominata "tabella di verità". Descrive graficamente i risultati delle operazioni logiche su ogni bit degli operandi. Ogni possibile combinazione di bit è inclusa nella tabella.

Si noti che i due operatori XOR ed EQV sono l'uno l'esatto contrario dell'altro.

A	NOT A	A	B	A AND B	A OR B	A XOR B	A EQV B	A IMP B
1	Ø	1	1	1	1	Ø	1	1
Ø	1	1	Ø	Ø	1	1	Ø	Ø
		Ø	1	Ø	1	1	Ø	1
		Ø	Ø	Ø	Ø	Ø	1	1

Tabella 6-1 Tabella di verità

Livello di Priorità degli Operatori Logici

Nello svolgimento di un'espressione vengono prima eseguite le operazioni numeriche e di confronto, e successivamente quelle logiche.

La tabella che segue riporta gli operatori logici nell'ordine di priorità in cui essi vengono svolti dal linguaggio BASIC.

OPERATORI	LIVELLO DI PRIORITA'
NOT	IL PIU' ALTO
AND	
OR	
XOR	
IMP	
EQV	IL PIU' BASSO

Tabella 6-2 Priorità degli Operatori Logici

Esempi

Consideriamo alcuni esempi. Prima di tutto assegneremo dei valori alle variabili X, Y e Z.

VIDEO	COMMENTI
X%=Ø Ok Y%=3 Ok Z%=5 Ok	il BASIC esegue le assegnazioni indicate
?X%<Y% AND Z%=3 Ø Ok	il risultato è falso (Ø), poichè X%<Y% è vero (-1) ma Z%=3 è falso (Ø)
?X% OR X%<Z% -1 Ok	il risultato è vero (-1), poiché X% è falso (Ø) ma X%<Z% è vero (-1)

```
?63 AND 16
 16
OK
```

il risultato è 16, poiché

```
63 = 111111 (in binario)
16 = Ø1ØØØØ (in binario)
     Ø1ØØØØ
```

```
?4 OR 2
 6
Ok
```

il risultato è 6, poiché

```
4 = 1ØØ (in binario)
2 = Ø1Ø (in binario)
    11Ø
```

```
?-1 OR -2
-1
Ok
```

il risultato è -1, poiché

```
-1 = 1111111111111111 (in binario)
-2 = 111111111111111Ø (in binario)
     1111111111111111
```

```
?Ø<2 AND 4=4
-1
Ok
```

il risultato è vero (-1), poichè

Ø<2 è vero (-1), e
4=4 è vero (-1)

```
?Ø XOR Y%=3
-1
Ok
```

il risultato è vero (-1), poiché

Ø è falso (Ø), e
Y%=3 è vero (-1)

```
?Z%>Y% AND NOT "A">"B"
-1
Ok
```

il risultato è vero (-1), poiché

Z%>Y% è vero (-1), e
"A">"B" è falso (Ø)

Nota: E' possibile scrivere due operatori di seguito solo a condizione che il secondo sia un operatore NOT.

Uso delle Espressioni Logiche

Le espressioni logiche possono essere usate per:

- controllare una condizione nelle seguenti istruzioni di controllo:

IF...GOTO...ELSE,
IF...THEN...ELSE,
WHILE

Ad esempio, l'istruzione

```
50 IF A$>B$ AND B<=C THEN 300
```

trasferirà il controllo dell'esecuzione all'istruzione 300 se la condizione (A$>B$ AND B<=C) è vera (cioè A$ maggiore di B$ e B minore o uguale di C). Se la condizione è falsa (cioè A$ minore di B$ o B maggiore di C) verrà eseguita l'istruzione successiva.

- controllare parole (16 bit) per una particolare configurazione di bit. Ad esempio, l'operatore AND può essere usato per "mascherare" tutti i bit eccetto uno di una status word (parola di stato) associata a un buffer di I/O. L'operatore OR può essere usato per fare il "merge" di due parole in modo da creare un determinato valore binario, ad esempio:

  -1 AND 8 è 8
  -1 OR 8 è -1

  poichè

  -1 = 1111111111111111 (in binario)

  8 = 0000000000001000 (in binario)

LIVELLO DI PRIORITA' DEGLI OPERATORI

La seguente tabella elenca tutti gli operatori (numerici, stringa, di confronto e logici) nell'ordine di priorità con cui essi vengono eseguiti nel linguaggio BASIC.

OPERATORI		LIVELLO DI PRIORITA'
^	(elevamento a potenza)	IL PIU' ALTO
-	(cambiamento di segno)	
* /	(moltiplicazione e divisione)	

\	(divisione tra interi)
MOD	(divisione modulo)
+ -	(addizione e sottrazione)
+	(concatenazione di stringa)
Tutti gli operatori di confronto	
NOT	
AND	
OR	
XOR	
IMP	
EQV	IL PIU' BASSO

Tabella 6 - 3 Priorità degli Operatori

Note

- gli operatori inclusi nella stessa linea hanno lo stesso livello di priorità

- tutti gli operatori di confronto hanno lo stesso livello di priorità

- l'ordine di svolgimento delle espressioni può essere modificato mediante l'uso di parentesi. Ad esempio, l'ordine di svolgimento di:

  NOT A>B AND C>D OR E>F

  è diverso da quello di:

  NOT (A>B AND (C>D OR E>F))

  se ad esempio, A > B è vero, C > D è falso ed E>F è vero, la prima espressione è vera, mentre la seconda è falsa.

- il risultato di una qualsiasi espressione può anche essere un operando, così si possono formare espressioni molto complesse, ad esempio concatenando due o più espressioni logiche con un operatore logico

(come nell'esempio di sopra). Comunque, non è buona norma di programmazione scrivere espressioni troppo complesse.

7. COME VENGONO EMESSI I DATI IN BASIC

SOMMARIO

Abbiamo appena analizzato come introdurre dati nell'M20 e come elaborarli.

Questo capitolo si propone di illustrare come definire l'ampiezza della linea dello schermo e/o della stampante (comando WIDTH) e come ottenere i risultati elaborati dal computer. Esamineremo le istruzioni LPRINT, PRINT, LPRINT USING e PRINT USING. Esse consentono di stampare/visualizzare i dati in formato standard o in formato definito dall'utente.

INDICE

DEFINIZIONE DEL NUMERO DI NULL E DELL'AMPIEZZA DELLA LINEA DI OUTPUT

Le istruzioni NULL e WIDTH, che possono anche essere utilizzate in un programma, consentono di stabilire rispettivamente il numero di null stampati dopo ogni linea e l'ampiezza massima della linea del video e della stampante.

Null è il primo carattere della tabella ASCII: non ha un equivalente grafico e corrisponde al codice decimale Ø (v. Appendice A).

NULL (PROGRAMMA/IMMEDIATO)

Stabilisce il numero di null che devono essere stampati alla fine di ogni linea.

Figura 7-1 Comando NULL

Esempio

SE l'utente imposta...	ALLORA...
NULL 2 CR	alla fine di ogni linea saranno stampati 2 null Nota: L'espressione numerica viene arrotondata al numero intero più vicino. Nel caso di nastri perforati a 1Ø caratteri per secondo, il valore dell'espressione dovrebbe essere >=3. Ciò consente anche di identificare le linee sul nastro.

Il valore dovrebbe essere Ø oppure 1 nel caso di telescriventi teletype o CRT teletype compatibile. Il valore dovrebbe essere uguale a 2 oppure a 3 per stampanti hardcopy da 3Ø caratteri per secondo.

WIDTH (PROGRAMMA/IMMEDIATO)

Definisce l'ampiezza massima della linea del video o della stampante, per gli output ottenuti usando le istruzioni PRINT/WRITE, LPRINT, PRINT USING e LPRINT USING o quando viene emesso un messaggio d'errore.

Figura 7-2 Comando WIDTH

Valore di Default

Quando non si usa il comando WIDTH il sistema assume un'ampiezza di linea dello schermo di 64 caratteri.

Allo stesso modo, se non si usa il comando WIDTH LPRINT, il sistema assume un'ampiezza di linea della stampante di 132 caratteri.

Esempio

```
1Ø PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
WIDTH 18
Ok
RUN
ABCDEFGHIJKLMNOPQR
STUVWXYZ
Ok
```

Caratteristiche

SE...	ALLORA...
si omette l'opzione LPRINT	si definisce l'ampiezza della linea del video
si fa uso del comando LPRINT	si definisce l'ampiezza della linea di stampa. Per esempio: 10 WIDTH LPRINT 4 20 LPRINT "AAAABBBBCC" RUN Ok AAAA BBBB CC
il valore dell'espressione numerica non è un numero intero	viene arrotondato all'intero più vicino e deve avere un valore compreso tra 15 e 255. Se il valore arrotondato è 255, l'ampiezza della linea è "infinita" e cioè il BASIC non esegue il ritorno a capo. Nondimeno la posizione del cursore e della testina stampante determinata dalle funzioni POS e LPOS ritorna a zero quando si supera la posizione 255. Se il valore arrotondato è maggiore di 255, viene emesso un messaggio d'errore (Illegal function call)

FORMATO STANDARD

Le istruzioni PRINT, WRITE e LPRINT consentono di visualizzare (PRINT e WRITE) oppure stampare (LPRINT) i risultati dei calcoli in un formato standard. Esse sono importanti istruzioni di programma, ma possono anche essere usate come istruzioni immediate.

Se si desidera includere in una linea i risultati di due o più espressioni, queste ultime devono essere separate da una virgola (WRITE), da una virgola o da un punto e virgola (PRINT, LPRINT).

Con l'istruzione WRITE i dati visualizzati verranno separati, l'uno dall'altro, da una virgola, mentre le stringhe saranno delimitate da virgolette. Con le istruzioni PRINT e LPRINT, l'uso della virgola "separa" i risultati (mediante l'introduzione di spazi), mentre il punto e virgola li "ravvicina" e le stringhe non saranno più delimitate da virgolette.

LPRINT/PRINT (PROGRAMMA/IMMEDIATO)

Le istruzioni LPRINT e PRINT consentono rispettivamente di stampare e visualizzare i dati secondo un formato standard.

E' possibile usare un punto interrogativo (?) al posto dell'istruzione PRINT.

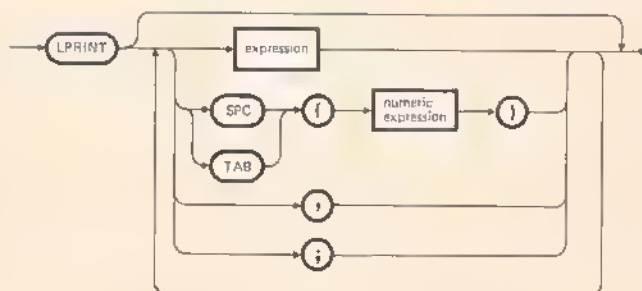



Figura 7-3 Istruzione LPRINT

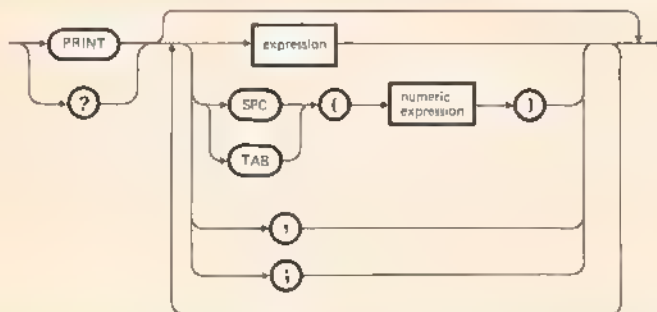



Figura 7-4 Istruzione PRINT

Caratteristiche

SE...	ALLORA...
l'istruzione LPRINT (oppure PRINT) non termina con una virgola o un punto e virgola	quando l'istruzione viene eseguita, viene generata una successiva linea di dati. Ad esempio: LIST 1Ø PRINT 1 2Ø PRINT 2 Ok RUN 1 2 Ok
l'istruzione LPRINT (oppure PRINT) non contiene alcuna espressione	viene saltata una linea, consentendo in questo modo di generare spazi tra linee di dati. Ad esempio: LIST 1Ø PRINT 1 2Ø PRINT 3Ø PRINT 2

```
Ok
RUN
 1

 2
Ok
```

le espressioni nella lista di output sono separate da una virgola

ogni valore viene stampato (o visualizzato) allineato a sinistra in una delle "zone" di stampa in cui ogni linea è divisa (ogni zona è composta da 14 posizioni).

Ad esempio:

```
LIST
10 A$ = "For June..."
20 X  = .353
30 PRINT "Results", A$, X
Ok
RUN
Results       For June...    .353
Ok
```

Nota: Ad ogni valore positivo (in questo caso .353) viene anteposto uno spazio (vedere in seguito).

Si noti anche che il numero delle zone di stampa dipende dal numero massimo di caratteri che può contenere. Ciò può essere stabilito mediante il comando WIDTH o assunto per default.

I valori stringa visualizzati (o stampati con LPRINT) non sono delimitati da virgolette.

la lista delle espressioni contiene molti dati

possono essere generate due o più linee di dati.

Ad esempio:

```
WIDTH 31
Ok
PRINT 1, 1+2, 2+3, 7, 9, "ABCD"
 1             3
 5             7
 9            ABCD
Ok
```

Nota: Ad ogni valore positivo viene anteposto uno spazio (vedere seguito)

un'istruzione LPRINT (oppure PRINT) contiene una o più espressioni numeriche

- ad ogni valore stampato o visualizzato viene aggiunto in coda uno spazio
- ad ogni valore positivo è anteposto uno spazio
- ad ogni valore negativo viene anteposto il segno meno
- ogni valore in semplice precisione viene rappresentato in formato a virgola fissa se la rappresentazione in tale formato può avvenire con 6 oppure meno di 6 cifre con la stessa accuratezza del formato a virgola mobile (o esponenziale)
- ogni valore in doppia precisione viene rappresentato in formato a virgola fissa se la rappresentazione in tale formato può avvenire con 15 oppure meno di 15 cifre con la stessa accuratezza del formato a virgola mobile (o esponenziale)

  Ad esempio:

```
PRINT 1Ø^-6
 .ØØØØØ1
Ok
PRINT 1Ø^-7
 1E-Ø7
Ok
PRINT 1D-15, 1D-16
 .ØØØØØØØØØØØØØØ1             1D-16
Ok
```

Nota: il secondo valore viene visualizzato allineato a sinistra nella terza zona di stampa poichè il primo valore supera l'ampiezza della prima zona di stampa

l'ultima espressione della lista è seguita da una virgola

una successiva operazione di PRINT (LPRINT) o INPUT visualizza o stampa il successivo carattere o cifra nella stessa linea (all'inizio della successiva zona di stampa), sempre che ci sia

spazio sufficiente; in caso contrario, esso sarà stampato o visualizzato nella linea successiva.

Ad esempio:

```
LIST
10 A$ = "For July..."
20 X = .491
30 PRINT "Results", A$,
40 PRINT X
Ok
RUN
Results       For July...    .491
```

l'ultima espressione della lista è seguita da un punto e virgola

una successiva operazione di PRINT/LPRINT o INPUT visualizza o stampa il successivo carattere o cifra nella stessa linea iniziando dalla posizione del cursore (o della testina di stampa) sempre che ci sia spazio sufficiente sulla linea. In caso contrario, esso sarà visualizzato/stampato nella linea successiva.

Ad esempio:

```
LIST
10 INPUT X
20 PRINT X "SQUARED IS" X^2 "AND";
30 PRINT X "CUBED IS" X^3
40 PRINT
50 GOTO 10
Ok
RUN
? 9
 9 SQUARED IS 81 AND 9 CUBED IS 729

? 21
 21 SQUARED IS 441 AND 21 CUBED IS 9261

?
```

si inseriscono più virgole in successione

l'uso di ogni virgola fa sì che la testina di stampa o il cursore si posizioni all'inizio della successiva zona di stampa.

L'uso delle virgole consente quindi di ottenere

una ampia spaziatura nella visualizzazione o stampa dei dati.

Ad esempio:

```
PRINT "M",,"N"
M                             N
Ok
```

si usano punti e virgola o spazi al posto delle virgole per separare le espressioni della lista

la spaziatura dei valori visualizzati o stampati è più compatta. L'esatta spaziatura dipende dal numero di cifre o caratteri di ogni valore. L'uso di punti e virgola consente quindi di stampare o visualizzare un numero maggiore di valori in ogni linea.

L'introduzione di più di uno spazio (o punto e virgola) tra due espressioni, avrà lo stesso effetto dell'introduzione di un solo spazio (o di un solo punto e virgola).

Ad esempio:

```
LIST
10 A1 = 1000
20 A2 = 2000
30 A3 = 3000
40 A4 = 4000
50 A5 = 5000
60 A6 = 6000
70 A7 = -7000
80 PRINT A1;A2;A3;A4;;A5 A6 A7
Ok
RUN
1000  2000  3000  4000  5000  6000 -7000
```

Gli spazi, che compaiono tra i numeri, sono dovuti al fatto che il sistema aggiunge uno spazio dopo ogni numero ed elimina il segno implicito "più" che precede ogni valore positivo, sostituendolo con uno spazio.

si introducono contemporaneamente virgole e punti e virgola nella stessa istruzione LPRINT (oppure PRINT)	è possibile identificare ogni risultato e ottenere una spaziatura appropriata nell'ambito di una linea. Ad esempio: LIST 1Ø INPUT "Base e Altezza"; B,H 2Ø PRINT "Area=";B*H,"Base=";B,"Altezza=";H 3Ø GOTO 1Ø Ok RUN Base e Altezza? 1.2,3 Area= 3.6 Base= 1.2 Altezza= 3 Base e Altezza? ∧ C Break in 1Ø Ok
si usa la funzione speciale di sistema TAB	si determina l'esatta posizione sulla linea in cui si dovrà posizionare la testina di stampa (o il cursore) per iniziare la stampa o la visualizzazione dei dati. Ad esempio: PRINT 1; TAB(6); 2 1 2 Ok
si usa la funzione speciale di sistema SPC	viene inserito il numero specificato di spazi sulla linea. (Nel calcolare il numero di spazi desiderati, bisogna tener conto che i dati numerici sono sempre seguiti da uno spazio). Ad esempio: PRINT 1; SPC(6); 2 1 2 Ok

WRITE (PROGRAMMA/IMMEDIATO)

Visualizza una lista di dati. Ogni elemento visualizzato è separato dal precedente da una virgola. Le stringhe sono delimitate da virgolette

("). Il BASIC esegue un ritorno a capo/interlinea dopo la visualizzazione dell'ultimo dato.

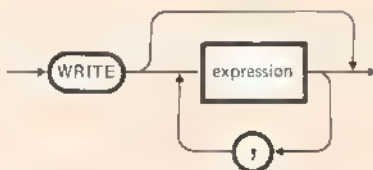



Figura 7-5 Istruzione WRITE

Dove

l' espressione può essere numerica, di confronto, logica o stringa. Se non si indica una espressione, l'istruzione WRITE esegue una interlinea.

Esempio

VIDEO	COMMENTI
1Ø A=8Ø : B=9Ø 2Ø C$="THAT'S ALL" 3Ø WRITE A,B,C$ RUN 8Ø, 9Ø, "THAT'S ALL" Ok	quando si esegue un'istruzione WRITE ogni dato è separato da quello precedente da una virgola, e le stringhe sono racchiuse tra virgolette. Nota: L'istruzione WRITE visualizza valori numerici usando lo stesso formato dell'istruzione PRINT, ma tali valori non saranno seguiti da spazi.

FORMATO DEFINITO DALL'UTENTE

Come abbiamo già visto, l'uso di virgole, punti e virgola, stringhe tra virgolette e delle funzioni SPC e TAB consente un controllo limitato del formato dei dati stampati o visualizzati. Le istruzioni LPRINT USING e PRINT USING consentono invece di avere un completo controllo del formato dei dati stampati o visualizzati.

Generalmente tali istruzioni sono utilizzate in un programma ma possono anche essere usate come istruzioni immediate.

LPRINT USING/PRINT USING (PROGRAMMA/IMMEDIATO)

Le istruzioni LPRINT USING e PRINT USING consentono rispettivamente di stampare e di visualizzare una lista di dati secondo un formato definito dall'utente.

Le espressioni che compaiono in queste istruzioni possono essere separate da una virgola (,) o da un punto e virgola (;). Non è rilevante se si usa la virgola o il punto e virgola come separatore. I valori saranno stampati o visualizzati secondo il formato specificato dall'espressione stringa che compare dopo USING. Questa espressione può essere una costante o una variabile stringa contenente speciali caratteri di formattazione. Tali caratteri determinano il campo ed il formato delle stringhe o dei numeri stampati o visualizzati.

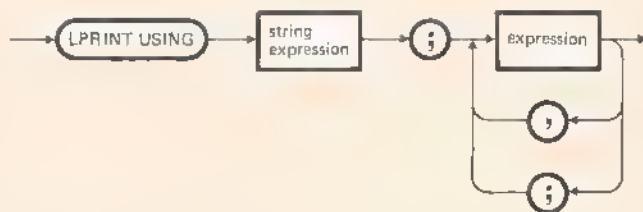



Figura 7-6 Istruzione LPRINT USING

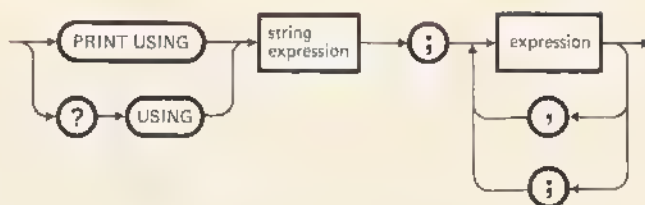



Figura 7-7 Istruzione PRINT USING

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string expression	stringa di caratteri di formattazione (vedere in seguito) o variabile stringa, comprendente una stringa di caratteri di formattazione
expression	espressione numerica, di confronto, logica o stringa che deve essere stampata o visualizzata

Visualizzazione o stampa di stringhe

E' possibile usare uno dei tre seguenti caratteri di formato:

CARATTERE DI FORMATO	SIGNIFICATO
"!"	Specifica che solo il primo carattere di una stringa deve essere stampato o visualizzato. Ad esempio: LIST 1Ø A$="WATCH" 2Ø B$="OUT"

```
30 PRINT USING "!";A$;B$
Ok
RUN
WO
Ok
```

"\n spazi\"

specifica che i primi 2+n caratteri della stringa devono essere stampati o visualizzati. Se le barre rovesce non contengono spazi, verranno stampati o visualizzati due caratteri. Se contengono uno spazio, i caratteri stampati (visualizzati) saranno tre, e così via. Se la stringa è più lunga del campo, i caratteri eccedenti saranno ignorati. Se invece il campo è più lungo della stringa, la stampa (visualizzazione) della stringa inizierà dalla sinistra e sarà completata con spazi sulla destra.

Ad esempio:

```
LIST
10 A$="LOOK"
20 B$="OUT"
30 PRINT USING "\  \";A$;B$
40 PRINT USING "\   \";A$;B$
Ok
RUN
LOOKOUT
LOOK OUT
```

"&"

specifica un campo stringa a lunghezza variabile. Quando il campo viene specificato con "&" la stringa verrà stampata esattamente come essa viene introdotta.

Ad esempio:

```
LIST
10 A$="LOOK":B$="OUT"
20 PRINT USING "!";A$;
30 PRINT USING "&";B$
Ok
RUN
LOUT
Ok
```

Visualizzazione e Stampa di Numeri

E' possibile usare i seguenti caratteri di formato:

CARATTERE DI FORMATO	SIGNIFICATO
#	un simbolo di numero ("number sign") viene usato per rappresentare la posizione di ogni cifra. Le posizioni di cifra sono sempre sostituite da cifre o da spazi. Se le cifre che devono essere stampate/visualizzate sono inferiori al numero delle posizioni specificate, il numero sarà allineato a destra e sarà preceduto da spazi.

Ad esempio:

```
PRINT USING "####";99 CR
  99
Ok
```

è possibile inserire un punto decimale in ogni posizione del campo. Se la stringa di formato indica che una cifra precede il punto decimale, tale cifra sarà stampata/visualizzata solo se diversa da zero. Quando risulti necessario, i numeri sono arrotondati.

Ad esempio:

```
PRINT USING "###.##";.78 CR
   .78
Ok

PRINT USING "###.##";987.654 CR
987.65
Ok

PRINT USING "##.##   ";10.,5.3,66.789,.234 CR
10.00    5.30   66.79     .23
Ok
```

Nell'ultimo esempio, alla fine della stringa di formato sono stati inseriti tre spazi per separare i valori visualizzati nella linea.

+ Un segno più all'inizio o alla fine della stringa di formato farà sì che il segno del numero (più o meno) venga visualizzato/stampato prima o dopo il numero.

Ad esempio:

```
PRINT USING "+##.##   ";-68.95,2.4,55.6,-.9 CR
-68.95    +2.40   +55.60     -.90
Ok
```

Nota: Per la sola visualizzazione del segno meno (e non del segno più), all'inizio del numero si dovrà iniziare la stringa di formattazione inserendo un simbolo di numero (#) in più.

Ad esempio:

```
PRINT USING "###.##";-68.95,68.95 CR
-68.95 68.95
```

\- un segno meno alla fine del campo di formato farà sì che i numeri negativi stampati/visualizzati siano seguiti dal segno meno.

Ad esempio:

```
PRINT USING "##.##-   ";-68.95,22.449,-7.01 CR
68.95-   22.45     7.01-
Ok
```

** un doppio asterisco all'inizio della stringa di formato fa sì che gli spazi iniziali di un campo numerico siano occupati da asterischi. Il simbolo ** indica anche due ulteriori posizioni di cifra.

Ad esempio:

```
PRINT USING "**#.#   ";12.39,-0.9,765.1 CR
*12.4   **-.9   765.1
Ok
```

$$

Un doppio segno di dollaro fa sì che un segno di dollaro compaia alla immediata sinistra del numero formattato. Il segno $$ specifica due ulteriori posizioni di cifra, una delle quali è occupata dal segno di $. Il carattere $$ non può essere usato con il formato esponenziale(^^^^) o con numeri negativi a meno che l'ultimo carattere della stringa di formato sia un segno meno (appare il segno del numero sulla destra se negativo) o un segno più (appare sempre il segno del numero sulla destra).

Ad esempio:

```
PRINT USING "$$###.##-";-456.78 CR
 $456.78-
Ok
```

**$

il segno **$ situato all'inizio di una stringa di formato combina gli effetti dei due simboli appena descritti. Gli spazi iniziali saranno occupati da asterischi, ed un segno di $ verrà inserito immediatamente prima del numero.

Il simbolo **$ specifica tre ulteriori posizioni di cifra.

Ad esempio:

```
PRINT USING "**$##.##";2.34 CR
***$2.34
Ok
```

,

una virgola collocata alla sinistra del punto decimale in una stringa di formato fa sì che una virgola venga visualizzata/stampata alla sinistra di ogni terza cifra nell'ambito della parte intera del numero. Una virgola situata alla fine della stringa di formato viene visualizzata/stampata come parte della stringa. Il simbolo della virgola mette a disposizione un'altra posizione di cifra. L'inserimento della virgola non esercita alcun effetto nell'ambito di un formato esponenziale (^^^^).

Ad esempio:

```
PRINT USING "####,.## ";1234.5 CR
1,234.5Ø
Ok
```

```
PRINT USING "####.##,";1234.5 CR
1234.5Ø,
Ok
```

^^^^

il formato esponenziale può essere specificato inserendo quattro simboli dell'esponente dopo i caratteri che specificano la posizione delle cifre. Tali simboli permettono di generare lo spazio per la visualizzazione/stampa di E+xx. E' possibile specificare qualsiasi posizione del punto decimale.

Le cifre significative sono allineate a sinistra e l'esponente viene modificato in consequenza. A meno che venga specificato un segno iniziale di + oppure un segno finale di + oppure di -,una posizione di cifra verrà utilizzata alla sinistra del punto decimale per stampare/visualizzare uno spazio oppure un segno meno.

Ad esempio:

```
PRINT USING "##.##^^^^";234.56 CR
 2.35E+Ø2
Ok
```

```
PRINT USING ".####^^^^";888888 CR
.8889E+Ø6
Ok
```

```
PRINT USING "+.####^^^^";123 CR
+.123ØE+Ø3
Ok
```

_

un segno di sottolineatura incluso nella stringa di formato fa sì che il successivo carattere sia visualizzato/stampato così come appare nella stringa di formato.

Ad esempio:

```
PRINT USING "_!##.##_!";12.34 CR
!12.34!
Ok
```

Il carattere visualizzato/stampato così come appare nella stringa di formato può consistere nello stesso segno di sottolineatura quando si include "_ _" nella stringa di formato.

%

se il numero che deve essere stampato/visualizzato è maggiore del campo numerico specificato, il segno di percentuale comparirà di fronte al numero. Se l'arrotondamento fa sì che il numero superi il campo, di fronte ad esso comparirà il segno di percentuale.

Ad esempio:

```
PRINT USING "##.##";111.22 CR
%111.22
Ok

PRINT USING ".##";.999 CR
%1.ØØ
Ok
```

Se il numero delle cifre specificate è superiore a 24, si avrà il messaggio d'errore "Illegal Function Call".

Nota

Se in un programma si usa più volte la stessa stringa di formato, sarà opportuno assegnare una varibile stringa dei caratteri di formattazione e quindi specificare il nome della variabile stringa al posto della stringa di formato.

Per esempio:

```
10 A$='  .  "
20 PRINT USING A$; 8.49
 .
 .
 .
100 PRINT USING A$;A,B,C
 .
 .
 .
150 PRINT USING A$;A1,B1
 .
 .
 .
RUN
 8.49
 0.00 0.00 0.00
 0.00 0.00
Ok
```

8. ISTRUZIONI DI CONTROLLO

SOMMARIO

Di norma le istruzioni BASIC sono eseguite l'una dopo l'altra nell'ordine del numero di linea. A volte però può essere necessario alterare la normale sequenza di esecuzione mediante "salti" (branches) che trasferiscono il controllo da un'istruzione ad un'altra parte del programma.

In questo capitolo vedremo come sia possibile conseguire questo risultato tramite salti condizionati o incondizionati e tramite cicli (loops).

INDICE

SALTI (TRASFERIMENTI) INCONDIZIONATI

I trasferimenti di controllo possono essere condizionati o incondizionati. L'istruzione GOTO determina un trasferimento incondizionato del controllo dell'esecuzione al numero di linea indicato nell'istruzione stessa. Il programma esemplificativo RETTANGOLO1 (vedere capitolo 1 e 2) contiene la seguente istruzione:

```
80 GOTO 20
```

L'istruzione indica all'M20 che la prossima istruzione che deve essere eseguita è la 20 al posto di quella immediatamente successiva di numero più elevato.

L'istruzione ON...GOTO, o istruzione calcolata GOTO, rappresenta un'altra forma di trasferimento incondizionato. Essa consente di trasferire il controllo ad una istruzione scelta tra n specificate in base al valore di una espressione numerica. Ad esempio:

```
100 ON A GOTO 15, 30, 500
```

Questa istruzione indica: se A=1 il controllo è trasferito all'istruzione 15, se A=2 all'istruzione 30, se A=3 all'istruzione 500. Se però A<1 oppure A>3 il programma BASIC continua con la prossima istruzione eseguibile.

GOTO (PROGRAMMA/IMMEDIATO)

Trasferisce il controllo dell'esecuzione alla linea di programma specificata.

Figura 8-1 Istruzione GOTO

Esempio:

VIDEO	COMMENTI
LIST 1Ø READ R 2Ø PRINT "R =";R, 3Ø A = 3.14*R^2 4Ø PRINT "AREA =";A 5Ø GOTO 1Ø 6Ø DATA 5,7,12 Ok RUN R = 5 AREA = 78.5 R = 7 AREA = 153.86 R = 12 AREA = 452.16 Out of data in 1Ø Ok	L'istruzione 5Ø trasferisce il controllo incondizionato all'istruzione 1Ø

Caratteristiche

SE...	ALLORA...
l'utente imposta: GOTO 5ØØ **CR** in Stato Comandi E 5ØØ è un'istruzione del programma residente in memoria	GOTO viene usato in alternativa a RUN. Nota: l'uso dell'istruzione GOTO in Stato Comandi fa sì che l'esecuzione del programma inizi al numero di linea specificato senza che venga automaticamente operato un CLEAR. Ciò consente di assegnare valori a variabili di programma quando si è in Stato Comandi. Questa tecnica può essere usata in fase di messa a punto (debug) del programma.
l'istruzione specificata tramite GOTO non è eseguibile (ad es. è una istruzione REM)	il controllo viene trasferito alla prima istruzione eseguibile successiva

un'istruzione GOTO si trova in un ciclo (loop) FOR/NEXT
E
trasferisce il controllo ad un'istruzione esterna al ciclo

il valore della variabile di controllo (vedere FOR/NEXT) è l'ultimo valore assunto nell'ambito del ciclo.

ON...GOTO (PROGRAMMA/IMMEDIATO)

Trasferisce il controllo dell'esecuzione di un programma ad una linea scelta tra un insieme di linee specificate dopo GOTO, in funzione del valore assunto da un'espressione specificata dopo ON.

Figura 8-2 Istruzione ON...GOTO

Caratteristiche

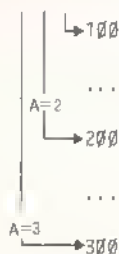
SE...

il programma ha la seguente struttura:

```
20 INPUT A
30 ON A GOTO 100,200,300
40
...
90
```

A=1

ALLORA...

il valore di A determina a quale dei numeri inclusi nella lista verrà trasferito il controllo. Se ad esempio il valore è uguale a tre, il controllo verrà trasferito al terzo numero di linea incluso nella lista (se il valore di A non è un intero, esso verrà arrotondato al valore intero più vicino).

il valore dell'espressione che segue ON è uguale a Ø o maggiore dei numeri inclusi nella lista (ma inferiore o uguale a 255)	il controllo viene trasferito alla prima istruzione eseguibile successiva
il valore dell'espressione che segue ON è negativo o superiore a 255	si ha il messaggio d'errore "Illegal Function Call" (Richiamo non Lecito di Funzione)

SALTI (TRASFERIMENTI) CONDIZIONATI

In alcuni casi si può desiderare di effettuare il trasferimento del controllo dell'esecuzione di un programma a parti diverse dello stesso programma a seconda che si verifichino condizioni predefinite. Le istruzioni IF...GOTO...ELSE e/o IF...THEN...ELSE possono essere utilizzate per verificare tali condizioni e per decidere come effettuare il trasferimento.

IF...GOTO...ELSE/IF...THEN...ELSE (PROGRAMMA/IMMEDIATO)

Entrambe le istruzioni effettuano il trasferimento condizionato ad una istruzione specificata.

Come è possibile notare attraverso la sintassi, l'istruzione IF... THEN...ELSE è la più potente, poichè consente di introdurre una serie di istruzioni sia dopo THEN che dopo ELSE.

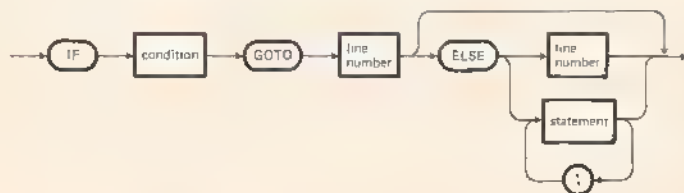



Figura 8-3 Istruzione IF...GOTO...ELSE

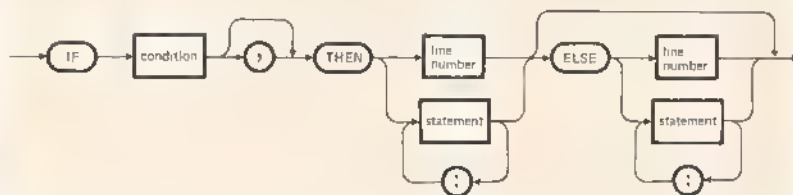



Figura 8-4 Istruzione IF...THEN...ELSE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
condition	può essere una espressione: - numerica - di confronto - logica Nota: Il BASIC determina se la condizione è vera o falsa controllando se il suo risultato (numerico) è diverso da zero (condizione vera) o uguale a zero (condizione falsa). Per questo motivo è possibile verificare se il valore di una variabile è uguale a zero o diverso da zero.

specificando semplicemente il nome della variabile come "condizione".

E' possibile inserire una virgola prima del THEN.

Caratteristiche

SE...	ALLORA...
la condizione è vera	il controllo viene trasferito O all'istruzione il cui numero di linea viene specificato dopo GOTO (o THEN) OPPURE alla prima istruzione specificata dopo THEN
la condizione è falsa E SE viene omessa la clausola ELSE	il controllo viene trasferito alla prossima istruzione eseguibile che segue l'istruzione IF...GOTO oppure IF...THEN
la condizione è falsa E SE la clausola ELSE è presente	il controllo viene trasferito O all'istruzione il cui numero di linea viene specificato dopo ELSE OPPURE alla prima istruzione specificata che segue ELSE. Nota: Dopo aver eseguito l'istruzione (o le istruzioni) che fanno seguito ad ELSE, il controllo viene trasferito alla prossima istruzione eseguibile

Esempi

VIDEO	COMMENTI
LIST 1Ø REM IF GOTO test program 2Ø INPUT X% 3Ø IF X%>=1Ø GOTO 6Ø 4Ø PRINT "IF GOTO failed the test" 5Ø GOTO 99 6Ø PRINT "IF GOTO passed the test" 99 GOTO 2Ø Ok RUN ? 1Ø IF GOTO passed the test ? 2 IF GOTO failed the test ? ^ C Break in 2Ø Ok	se si introduce 1Ø la condizione dell'istruzione 3Ø (X%>=1Ø) è vera e quindi il controllo viene trasferito all'istruzione 6Ø. Se si introduce 2 la condizione è falsa e quindi il controllo viene trasferito all'istruzione 4Ø

LIST
1Ø INPUT X
2Ø IF X=INT(X)
   THEN PRINT X; "is an integer"
   ELSE PRINT X; "is not an integer"
3Ø IF X=9999 THEN END ELSE 1Ø
Ok
RUN
? 1
 1 is an integer
? 1.5
 1.5 is not an integer
? ^ C
Break in 1Ø
Ok
	se si introduce 1, la condizione (X=INT(X)) indicata nell'istruzione 2Ø, è vera e quindi il controllo viene trasferito all'istruzione PRINT che segue THEN. Se invece si introduce 1.5 la condizione è falsa ed il controllo è trasferito all'istruzione PRINT che segue ELSE.
	
	Nota: l'istruzione 2Ø è una sola linea logica divisa in tre linee fisiche.

5Ø IF 1 THEN A=1ØØØ	il valore 1ØØØ viene assegnato alla variabile A se 1 non è uguale a zero

7Ø IF (I< 3Ø) AND (I >5) THEN A=B+C:GOTO 35Ø 8Ø PRINT "OUT OF RANGE" . . .	si controlla se il valore di I è maggiore di 5 e minore di 3Ø. Se I si colloca in quest'ambito, si calcola il valore della variabile A ed il controllo viene poi trasferito all'istruzione 35Ø. In caso contrario, l'esecuzione continua con la linea 8Ø

Istruzioni IF Nidificate (Nested)

Le istruzioni IF...GOTO...ELSE oppure IF...THEN...ELSE possono essere nidificate. Tale processo trova un limite solo nella lunghezza di una linea BASIC (255 caratteri). Ad esempio:

SE...	ALLORA...
si digita: IF X>Y THEN PRINT "GREATER" ELSE IF Y>X THEN PRINT "LESS THAN" ELSE PRINT "EQUAL" **CR**	la linea introdotta è una istruzione lecita (in questo caso è una sola linea logica divisa in tre linee fisiche)
l'istruzione non contiene lo stesso numero di clausole ELSE e THEN	ogni ELSE viene associato con il più prossimo THEN non ancora associato a una clausola ELSE. Ad esempio: 1ØØ IF A=B THEN IF B=C THEN PRINT "A=C" ELSE PRINT "A<>C" 11Ø... Verrà visualizzato A=C quando A=B e B=C; verrà vizzualizato A<>C quando A=B ma B diverso da C. Se A diverso da B si passa all'istruzione 11Ø.

Controllo di un Valore Espresso in Virgola Mobile

SE...	ALLORA...
si utilizza un'istruzione IF...GOTO...ELSE, oppure IF...THEN...ELSE per controllare il risultato di un calcolo in virgola mobile	poichè la rappresentazione interna del valore non può essere esatta, il controllo viene effettuato nell'ambito del grado di accuratezza definito. Per controllare ad esempio se la variabile A è uguale a 1.Ø si utilizzerà: IF ABS(A-1.Ø)<1.ØE-6 GOTO... oppure IF ABS(A-1.Ø)<1.ØE-6 THEN... Questo controllo risulterà vero se il valore di A è uguale ad 1.Ø con un errore relativo inferiore a 1.ØE-6

CICLI ITERATIVI (LOOPS)

L'esecuzione ripetuta di una serie di istruzioni viene definita con il termine di loop (ciclo iterativo).

Ciò può essere ottenuto mediante le istruzioni:

- FOR e NEXT, che possono essere usate per racchiudere una serie di istruzioni, consentendo di ripeterle un numero specificato di volte
- WHILE e WEND, che possono essere usate per racchiudere una serie di istruzioni, consentendo di ripeterle finchè una determinata condizione risulta vera.

Come un Loop può Semplificare il Vostro Problema

Si supponga di voler visualizzare una lista dei numeri da 1 a 25 contemporaneamente alla loro radice quadrata. Ciò può essere ottenuto con gradi di efficienza via via maggiori.

Il metodo più primitivo è quello di introdurre le seguenti istruzioni:

```
10 PRINT 1,SQR(1)
20 PRINT 2,SQR(2)
30 PRINT 3,SQR(3)
 .
 .
 .
```

e così via, per terminare con:

```
240 PRINT 24,SQR(24)
250 PRINT 25,SQR(25)
260 END
```

Una soluzione migliore può essere ottenuta mediante l'uso dell'istruzione IF...THEN

```
10 LET A=1
20 PRINT A,SQR(A)
30 LET A=A+1
40 IF A<26 THEN 20
50 END
```

E' possibile semplificare ulteriormente il processo usando il loop FOR/NEXT

```
10 FOR A=1 TO 25
20 PRINT A,SQR(A)
30 NEXT A
40 END
```

Quest'ultima semplificazione può sembrare a prima vista di scarso rilievo. Nondimeno le utilizzazioni di questo loop sono sorprendenti così come potrà risultare dall'esame di alcune possibilità del suo impiego che verranno presentate in seguito.

L'inizio del Loop - L'istruzione FOR

Le istruzioni FOR e NEXT identificano rispettivamente l'inizio e la fine di un loop. FOR specifica quante volte il loop, e cioè l'istruzione o la sequenza delle istruzioni comprese tra FOR e NEXT devono essere eseguite.

Nell'esempio precedente FOR specifica che l'istruzione PRINT deve essere eseguita per valori successivi di A compresi tra 1 e 25 (il valore di A viene incrementato di 1 dopo l'esecuzione di ogni PRINT). Quando il valore di A supera 25, l'esecuzione del loop termina, ed il controllo passa alla prima istruzione successiva all'istruzione NEXT. In questo caso si tratta dell'istruzione END, che sta ad indicare il completamento del programma.

L'indicazione A=1 TO 25 definisce l'insieme dei valori presi in considerazione per eseguire il loop. In questo contesto A è denominata variabile di controllo, poichè controlla quante volte il loop deve essere eseguito. La variabile di controllo subisce sempre incrementi pari ad 1, a meno che l'istruzione FOR non specifichi diversamente. E' possibile difatti incrementare la variabile di controllo di un valore diverso da 1. Ciò si ottiene aggiungendo la clausola STEP.

Ad esempio:

```
10 FOR A=1 TO 25 STEP 2
```

Questa istruzione indica che la variabile di controllo subirà un incremento (o step) di 2. Il loop verrà quindi eseguito per ogni valore dispari di A compreso tra 1 e 25 (e cioè 1,3,5...25). Quando il valore di A supera 25 (e cioè quando raggiunge 27) l'esecuzione del loop termina. Il valore di A sarà quindi 27 prima che venga eseguita l'istruzione che segue l'istruzione NEXT.

Se si desidera invece eseguire il loop per i valori pari di A compresi tra 1 e 25, ciò può essere ottenuto specificando:

```
10 FOR A=2 TO 25 STEP 2
```

Anche in questo caso quando il valore di A supera 25 (cioè raggiunge 26) l'esecuzione del loop termina.

E' possibile esplicitare, anche se non è necessario, un valore dell'incremento pari a 1. Ad esempio:

```
80 FOR X=1 TO 40 STEP 1
```

Come nel caso delle espressioni contenute nelle istruzioni LET e PRINT i parametri dell'istruzione FOR possono essere molto complessi. Ognuna delle seguenti istruzioni, ad esempio, è valida:

```
7Ø FOR A=B TO C
8Ø FOR X=8/M+N TO A^2
5Ø FOR I=SQR(A) TO 155Ø STEP B*C+6
```

Se il valore di un incremento è negativo, il ciclo FOR/NEXT è eseguito finchè il valore della variabile di controllo non è minore del valore finale (cioè il valore espresso dopo la parola TO).

Ad esempio:

```
LIST
1Ø FOR K%=1 TO -1Ø STEP -1
2Ø PRINT K%;
3Ø NEXT K%
Ok
RUN
 1  Ø  -1  -2  -3  -4  -5  -6  -7  -8  -9  -1Ø
```

In questo caso il ciclo viene ripetuto 12 volte.

Il Termine del Ciclo Iterativo - L'istruzione NEXT

Così come il loop inizia sempre con un'istruzione FOR, esso termina sempre con un'istruzione NEXT. Si ricordi che tale ciclo comprende tutte le istruzioni incluse tra le istruzioni FOR e NEXT.

L'istruzione NEXT inizia con la parola chiave NEXT, alla quale può seguire una lista di nomi di variabili di controllo. Ogni variabile di controllo nell'istruzione NEXT deve avere lo stesso nome della variabile di controllo della corrispondente istruzione FOR. Più istruzioni FOR possono essere associate a una sola NEXT (si veda in seguito Nested Loops (Cicli Nidificati)).

FOR/NEXT (PROGRAMMA/IMMEDIATO)

Le istruzioni FOR/NEXT consentono di eseguire in loop una serie di istruzioni per un certo numero di volte.

Figura 8-5 Istruzione FOR

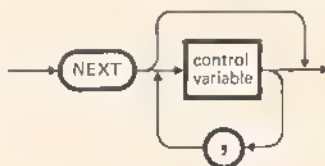



Figura 8-6 Istruzione NEXT

Dove:

ELEMENTO DI SINTASSI	SIGNIFICATO	VALORI DI DEFAULT
control variable	consiste in una variabile semplice numerica (intera o in semplice precisione). I nomi delle variabili di controllo (control variable) specificati nelle istruzioni NEXT e FOR devono essere uguali. Alla parola NEXT può seguire una lista di variabili di controllo (vedere Nested Loops) ma è anche possibile scrivere un'istruzione NEXT costituita da questa sola parola.	se alla parola NEXT non segue alcuna variabile di controllo, l'istruzione NEXT verrà messa in corrispondenza con l'istruzione FOR eseguita per ultima

initial value	consiste in un'espressione numerica che specifica il primo valore (cioè il valore iniziale) assegnato alla variabile di controllo quando si esegue una istruzione FOR	
final value	consiste in un'espressione numerica che specifica il valore limite della variabile di controllo. Detto valore viene confrontato con la variabile di controllo ogni volta che il ciclo viene ripetuto	
increment	consiste in un'espressione numerica che specifica l'incremento cioè il valore che deve essere aggiunto (col suo segno algebrico) alla variabile di controllo ogni volta che l'istruzione NEXT viene eseguita	se non viene specificata l'opzione STEP si assume un incremento di +1.

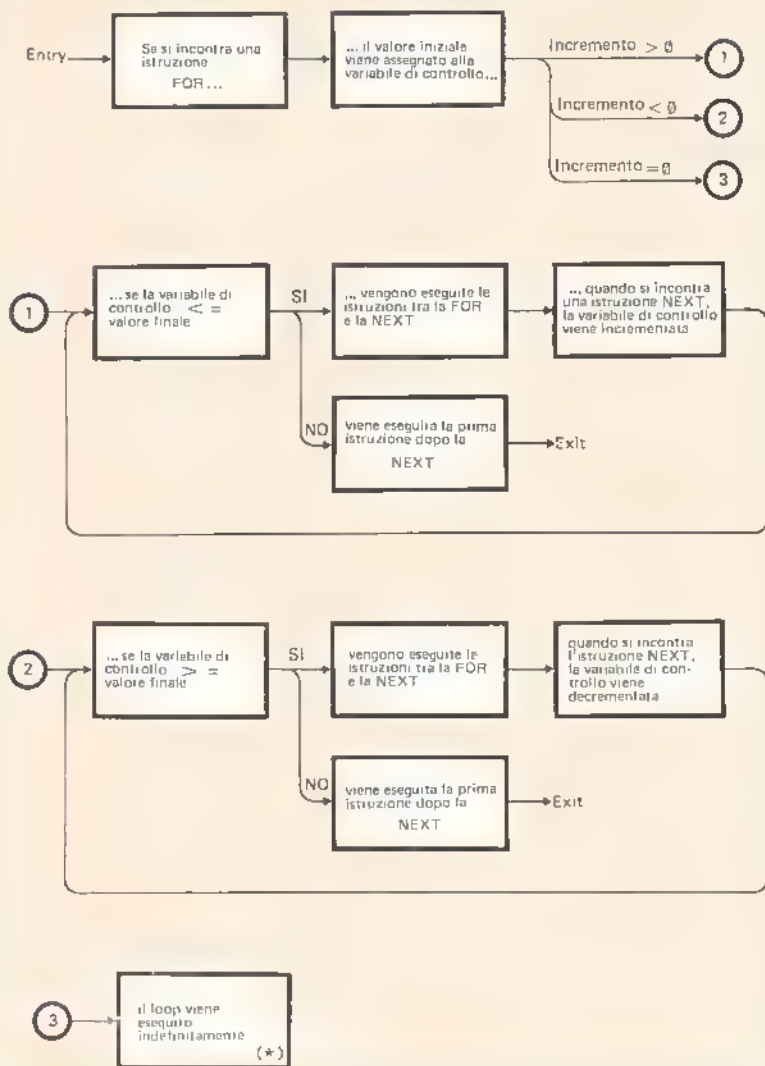
Funzionamento delle Istruzioni FOR/NEXT

(*) A meno che il valore iniziale e finale coincidano. In tal caso viene eseguita la prima istruzione dopo la NEXT

Figura 8-7 Istruzioni FOR/NEXT

Nota

Un ciclo FOR/NEXT può essere definito "pendente" (pending) se interotto da un "break" prima della sua conclusione. Qualsiasi modifica al programma residente (come ad esempio: cancellazione e modifica di linee e così via) impedirà la conclusione del ciclo.

Incremento di Valore Positivo

SE...	ALLORA...
il valore dell'incremento è positivo	il ciclo FOR/NEXT viene eseguito finchè il valore della variabile di controllo non è maggiore del valore finale. Ad esempio: LIST 1Ø K=1Ø 2Ø FOR I=1 TO 1Ø STEP 2 3Ø PRINT I; 4Ø K=K+1Ø 5Ø PRINT K 6Ø NEXT Ok RUN 1 2Ø 3 3Ø 5 4Ø 7 5Ø 9 6Ø Ok In questo caso il ciclo è ripetuto cinque volte

il valore dell'incremento è positivo
E SE
il valore iniziale è superiore a quello finale

il ciclo non viene eseguito.
Ad esempio:

```
LIST
10 J=0
20 FOR I=1 TO J
30 PRINT I
40 NEXT I
50 PRINT "Exit of the loop"
Ok
RUN
Exit of the loop
Ok
```

Incremento di Valore Negativo

SE...

il valore dell'incremento è negativo

ALLORA...

il ciclo viene eseguito finchè il valore della variabile di controllo non è minore del valore finale.

Ad esempio:

```
LIST
10 FOR I%=1 TO -10 STEP -3
20 PRINT I%;
30 NEXT I%
40 PRINT
50 PRINT "Exit ";
   "CONTROL VARIABLE=";I%
Ok
RUN

 1 -2 -5 -8
Exit CONTROL VARIABLE=-11
```

In questo caso il ciclo è eseguito quattro volte. Quando termina, la variabile di controllo mantiene l'ultimo valore assunto (-11) che viene visualizzato mediante l'istruzione 50

il valore dell'incremento è negativo E SE il valore iniziale è minore di quello finale	il ciclo non viene eseguito. Ad esempio: LIST 10 FOR K%=1 TO 10 STEP -2 20 PRINT K%; 30 NEXT K% 40 PRINT "Exit"; "CONTROL VARIABLE=";K% Ok RUN Exit CONTROL VARIABLE=1 Ok

Incremento di Valore Uguale a Zero

SE...	ALLORA...
il valore dell'incremento è uguale a zero	il ciclo viene ripetuto senza fine (a meno che il valore iniziale ed il valore finale coincidano, in questo caso il loop non verrà affatto eseguito) Ad esempio: LIST 100 FOR A%=1 TO 30 STEP 0 110 PRINT A%; 120 NEXT A% Ok RUN 1 1 1... Si dovrà impostare CTRL C per interrompere l'esecuzione

Nested Loops (Cicli Nidificati)

Due o più cicli FOR/NEXT possono essere nidificati a condizione che il ciclo FOR/NEXT interno sia interamente compreso in quello esterno. I seguenti cicli, ad esempio, sono corretti:

```
┌── 5Ø FOR I = 1 TO 1Ø
│┌─1ØØ FOR J = 2 TO 2Ø
│└─2ØØ NEXT J
└──3ØØ NEXT I
```

mentre i seguenti non lo sono:

```
┌── 5Ø FOR I = 1 TO 1Ø
│┌─1ØØ FOR J = 2 TO 2Ø
└┼─15Ø NEXT I
 └─2ØØ NEXT J
```

Due o più cicli FOR/NEXT nidificati non devono avere la stessa variabile di controllo.

Ad ogni istruzione FOR deve corrispondere una istruzione NEXT.

Se i cicli nidificati hanno lo stesso punto finale, si può utilizzare per tutti una sola istruzione NEXT con una lista di tutte le variabili di controllo separate da una virgola.

Esempio:

```
5Ø FOR I = 1 TO 1Ø
 .
 .
 .
1ØØ FOR J = 2 TO 2Ø
 .
 .
 .
2ØØ NEXT J,I
```

Quando si incontra un ciclo nidificato, questo viene eseguito. Alla fine dell'esecuzione del ciclo viene eseguita la prima istruzione successiva all'istruzione NEXT associata.

Non c'è limite al numero di cicli interni a un altro ciclo. Il numero di loop contemporaneamente attivi è limitato solo dalla quantità di memoria disponibile.

I cicli nidificati forniscono un metodo di programmazione molto utile per risolvere un'ampia gamma di problemi. Qui di seguito viene illustrato un esempio di ciclo nidificato.

Esempio

VIDEO	COMMENTI
LIST 10 REM PRIME NUMBERS 20 INPUT "Enter limits N,M";N,M 30 PRINT "Primes from";N;"TO";M 40 PRINT 50 PRINT 60 FOR I=N TO M 70 LET K=SQR(I) 80 FOR J=2 TO K 90 LET E=I/J-INT(I/J) 100 IF E=0 THEN 130 110 NEXT J 120 PRINT I; 130 NEXT I 140 PRINT 150 PRINT 160 PRINT "End of List" 170 END Ok RUN Enter limits N,M? 1,15 Primes from 1 TO 15 1 2 3 5 7 11 13 End of List Ok	sono visualizzati tutti i numeri primi compresi in un dato insieme. Un ciclo FOR/NEXT specifica l'insieme dei numeri presi in considerazione. Un secondo ciclo, nidificato nel primo, contiene un algoritmo per determinare quali numeri dell'insieme specificato siano un numero primo. Per spiegare l'algoritmo: i numeri assegnati ad una variabile (in questo caso I) sono divisi per un numero intero (in questo caso J) il cui valore è compreso tra 2 e la radice quadrata di I. Se il resto della divisione è uguale a 0, I non è un numero primo. Viene allora generato il numero I+1 e l'esecuzione viene ripetuta. La scelta della radice quadrata del valore finale viene effettuata poichè se vi sono fattori interi del numero I, essi saranno sempre collocati tra 2 e la radice quadrata di I. Nota: L'istruzione 100 consente di uscire dal ciclo interno anche se J non è maggiore di K. E' difatti possibile uscire da un ciclo quando una determinata condizione è soddisfatta. Non è invece possibile entrare nel "mezzo" di un ciclo.

Note

- se un'istruzione NEXT viene eseguita prima della corrispondente istruzione FOR, viene emesso il messaggio di errore:

  NEXT without FOR

  e l'esecuzione del programma viene interrotta.

  Ad esempio:

```
1200 IF A>5 THEN 2010
  .
  .
  .
2000 FOR J=1 TO 7
2010 PRINT "HELLO";
2020 NEXT J
```

  Quando si esegue l'istruzione 2020 a seguito di un salto dall'istruzione 1200, il BASIC visualizza il sopracitato messaggio di errore e ritorna in Stato Comandi.

- il valore finale è sempre definito prima di quello iniziale. Se ad esempio si introduce:

```
10 I=5
20 FOR I=1 TO I+5
```

  l'istruzione 20 assegnerà il valore 10 al valore finale. Comunque per agevolare la leggibilità del programma, non è consigliabile usare la variabile di controllo per definire il valore finale.

- se possibile, è buona norma di programmazione usare una variabile intera per la variabile di controllo e costanti (o variabili) intere per il valore iniziale e finale e per l'incremento. Ciò ottimizzerà la velocità di esecuzione.

WHILE/WEND (PROGRAMMA/IMMEDIATO)

Esegue in ciclo una serie di istruzioni finchè una determinata condizione è vera.

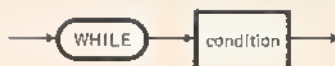



Figura 8-8 Istruzione WHILE

Figura 8-9 Istruzione WEND

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
condition	può consistere in una espressione: - numerica, - di confronto, - logica Nota: Il programma BASIC determina se la condizione (condition) è vera o falsa controllando se il risultato dell'espressione è diverso o uguale a zero. Nel primo caso il risultato viene considerato vero, nel secondo falso. Ciò permette di verificare se il valore di una variabile è diverso o uguale a zero, specificando semplicemente il nome della variabile come condizione.

Funzionamento delle Istruzioni WHILE/WEND

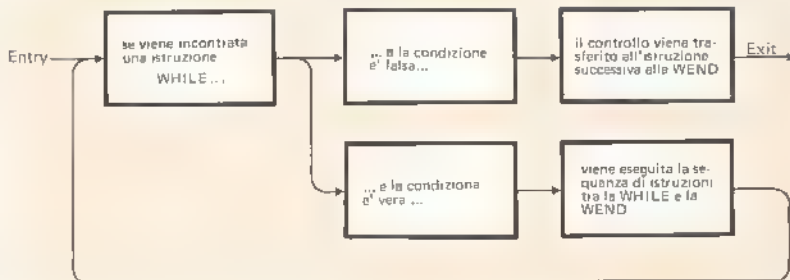



Figura 8-10 Istruzioni WHILE/WEND

Nota

Un ciclo WHILE/WEND può essere definito "pendente" (pending) se interrotto da un "break" prima della sua conclusione. Qualsiasi modifica al programma residente (come ad esempio: cancellazione o modifica di linee e così via) impedirà la conclusione del ciclo.

Esempio:

VIDEO	COMMENTI
LIST 90 'BUBBLE SORT ARRAY A$ 100 FLIPS=1 'FORCE ONE PASS THRU LOOP 110 WHILE FLIPS 115 FLIPS=0 120 FOR I=1 TO J-1 130 IF A$(I)>A$(I+1) THEN SWAP A$(I),A$(I+1):FLIPS=1 140 NEXT I 150 WEND Ok RUN Ok	gli elementi della matrice A$ vengono memorizzati in ordine ascendente (dall'indice 1 all'indice J) Nota: la condizione (in questo caso il valore della variabile FLIPS) può cambiare durante il ciclo (vedi istruzione 130).

Nota

Non c'è limite al numero di cicli WHILE/WEND interni a un altro ciclo WHILE/WEND. Ogni istruzione WEND verrà associata alla istruzione WHILE eseguita per ultima. Un'istruzione WHILE a cui non corrisponda un'istruzione WEND causa l'errore "WHILE without WEND", così come un'istruzione WEND a cui non corrisponda un'istruzione WHILE causa un errore "WEND without WHILE".

E' possibile uscire da un ciclo WHILE/WEND, o perchè la condizione specificata dopo WHILE è diventata falsa, o tramite un'istruzione IF...THEN o GOTO.
Non è invece possibile entrare in un ciclo WHILE/WEND senza eseguire l'istruzione WHILE.

9. FUNZIONI

SOMMARIO

Questo capitolo descrive le funzioni built-in (di sistema) che possono essere richiamate da ogni programma senza che sia richiesta una ulteriore definizione e le funzioni definite dall'utente che possono essere utilizzate esattamente nello stesso modo, ma solo nell'ambito del programma nel quale sono state definite.

INDICE

INTRODUZIONE

Vi sono delle circostanze in cui nell'ambito dello stesso programma si deve fare ricorso più volte ad una stessa espressione.

Per evitare di scrivere tali espressioni più di una volta e per risparmiare spazio è possibile definire funzioni e richiamarle in più punti del programma.

Il richiamo di una funzione si può ottenere specificando il suo nome seguito da uno o più argomenti tra parentesi, rappresentanti i valori da passare ai parametri. Ad ogni argomento corrisponde un parametro nella definizione della funzione.

Gli argomenti sono separati da virgole. Possono essere costanti, variabili o espressioni. Anche i parametri sono separati da virgole, ma possono essere solo delle variabili.

Il numero dei parametri deve essere uguale a quello degli argomenti e il tipo di ogni argomento (numerico o stringa) deve essere uguale al tipo del parametro corrispondente. La corrispondenza è di tipo posizionale (cioe al primo argomento corrisponde il primo parametro, ecc...). E' possibile trasferire uno o più argomenti a una funzione o addirittura nessuno.

Sono possibili conversioni nel passaggio di argomenti numerici a parametri di tipo diverso. Per esempio se si fornisce un valore a virgola mobile nel caso in cui è richiesto un intero, il BASIC arrotonderà tale valore all'intero più vicino.

Una funzione calcola un solo valore, che può essere di tipo numerico o stringa in funzione del tipo di espressione usata per definire la funzione.

Le funzioni del linguaggio BASIC possono essere classificate in due categorie principali:

- Funzioni built-in o di sistema

  Le funzioni di sistema rappresentano una parte intrinseca del linguaggio BASIC, e consentono di effettuare un insieme di operazioni di tipo numerico o stringa di uso comune. L'utente può utilizzarle in qualunque programma senza dover procedere ad una definizione esplicita. Un elenco completo delle funzioni di sistema corredato da una descrizione viene fornito di seguito.

- Funzioni definite dall'utente

Nell'ambito di un programma BASIC l'utente può definire un numero arbitrario di funzioni utilizzando l'istruzione DEF FN. Il nome di una funzione definita dall'utente inizia con FN e può essere qualunque nome valido di variabile.

Tutte le funzioni devono essere definite prima di essere richiamate.

Esempi:

VIDEO	COMMENTI
1Ø A=X*SIN(X)+LOG(X)	in questo caso SIN e LOG sono funzioni numeriche di sistema
LIST 1Ø DEF FNH(X,Y)=SQR(X*X+Y*Y) 2Ø INPUT "SIDES";X1,Y1 3Ø PRINT "H=";FNH(X1,Y1); "X1=";X1;"Y1=";Y1 4Ø GOTO 2Ø Ok RUN SIDES? 3.5,1.2 H= 3.7 X1= 3.5 Y1= 1.2 SIDES? 1.7,4 H= 4.34626 X1= 1.7 Y1= 4 SIDES? ^C Break in 2Ø Ok	FNH è una funzione definita dall'utente. Essa viene definita dall'istruzione DEF FN (vedere l'istruzione 1Ø), e calcola la radice quadrata della somma dei quadrati dei parametri X ed Y usando la funzione di sistema SQR. L'istruzione 3Ø richiama la funzione definita dall'utente attribuendo due argomenti ai parametri corrispondenti. Nota: I nomi degli argomenti possono non coincidere con quelli dei parametri corrispondenti.

FUNZIONI DEFINITE DALL'UTENTE

Se si deve utilizzare ripetutamente una equazione di tipo numerico o stringa è molto più conveniente definire la stessa sotto forma di funzione. La funzione così definita può essere richiamata esattamente

nello stesso modo delle funzioni di sistema. L'unico vincolo consiste nel fatto che la definizione è valida solo per quel determinato programma e deve quindi essere ridefinita in ogni programma che richiede la sua utilizzazione (a meno che il secondo programma non sia concatenato al primo mediante l'uso dell'opzione MERGE).

DEF FN (PROGRAMMA)

DEF FN definisce una funzione utente di tipo numerico o stringa.

L'istruzione DEF FN deve essere eseguita prima che la funzione che essa definisce possa essere richiamata.

L'istruzione DEF FN non é ammessa nel calcolo immediato.

Figura 9-1 Istruzione DEF FN

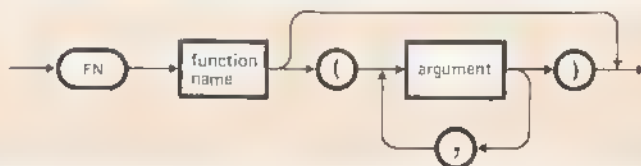



Figura 9-2 Richiamo della Funzione

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
function name	un nome valido di variabile che inizia con FN. Non si devono inserire spazi tra FN e il nome della funzione e il primo carattere del nome deve essere una lettera.

	Se il nome della funzione specifica anche un tipo, il valore della funzione viene ricondotto a quel tipo.
parameter	una variabile "dummy" che dovrà essere rimpiazzata dal valore dell'argomento corrispondente quando la funzione viene richiamata. L'associazione argomenti/parametri è di tipo posizionale (cioè al primo argomento corrisponde il primo parametro e cosi via).
argument	il valore effettivo che dovrà essere attribuito al parametro corrispondente. Ogni argomento può essere una costante, una variabile o una espressione.
expression	una espressione che esegue l'operazione della funzione. Il tipo dell'espressione (numerico o stringa) deve coincidere con quello della funzione. L'espressione include di norma come variabili solo i parametri della funzione, ma può includere anche variabili di programma definite esternamente alla definizione della funzione (variabili globali). I nomi di parametri che compaiono nell'espressione servono unicamente a definire la funzione e non influenzano eventuali variabili di programma che hanno lo stesso nome. Tuttavia, per una maggiore leggibilità del programma, si sconsiglia di dare lo stesso nome a un parametro e a una variabile di programma.

Caratteristiche

SE...	ALLORA...
il tipo di un argomento non ha le stesse caratteristiche del parametro corrispondente	si verifica un errore di "type mismatch" (incompatibilità di tipo)

una funzione definita dall'utente viene richiamata prima di essere definita	si verifica l'errore di "Undefined user function" (Funzione utente non definita)

una funzione definita dall'utente viene richiamata da un'altra funzione definita dall'utente	la funzione richiamata deve essere definita nell'ambito dello stesso programma in una posizione precedente.
	
	Ad esempio:
	
	`10 DEF FNA(X)=(SIN(X/5)*3.1)/180`
	`20 DEF FNB(X)=(FNA(X)+SIN(X))*.5`

un programma concatena un altro con l'opzione MERGE	le definizioni di funzione devono essere scritte nel programma concatenante prima dell'istruzione CHAIN se vengono richiamate dal programma concatenato. Altrimenti, quando verrà completata l'operazione di MERGE, la funzione definita dall'utente risulterà non definita. (per maggiori dettagli vedere il capitolo 11).
	
	Ad esempio:
	
	`10 DEF FNA(X)=(X+X*(X+1))`
	`.`
	`.`
	`.`
	`100 CHAIN MERGE "V1:PROG1"`

Note

La sintassi del Richiamo di Funzione è valida sia per le funzioni definite dell'utente sia per le funzioni di sistema.

FUNZIONI NUMERICHE DI SISTEMA

Il linguaggio BASIC fornisce un numero di routine standard che consentono di evitare di scrivere gruppi di istruzioni per calcolare funzioni matematiche quali la radice quadrata, il seno o il logaritmo naturale ecc... Ad eccezione della funzione CDBL che ritorna un risultato in doppia precisione, i risultati delle funzioni numeriche di sistema sono o numeri interi o numeri in singola precisione.

Tutte le funzioni numeriche di sistema (built-in) sono elencate qui di seguito in ordine alfabetico.

Nota: In questo paragrafo descriveremo anche l'istruzione RANDOMIZE, perchè strettamente connessa alla funzione RND.

ABS (PROGRAMMA/IMMEDIATO)

Calcola il valore assoluto di una espressione numerica.

Figura 9-3 Funzione ABS

Esempio

```
PRINT ABS (7*(-5))
 35
Ok
```

ATN (PROGRAMMA/IMMEDIATO)

Calcola l'arcotangente dell'argomento.

Il valore calcolato è espresso in radianti e si colloca nell'ambito di $-\pi/2$ e $\pi/2$ (dove π è 3.1415...)

Figura 9-4 Funzione ATN

Esempio

```
10 INPUT X
20 PRINT ATN(X)
Ok
RUN
? 3
 1.24905
Ok
```

Note

Il calcolo di ATN viene eseguito in semplice precisione.

CDBL (PROGRAMMA/IMMEDIATO)

La funzione CDBL converte qualsiasi formato numerico in un argomento in doppia precisione (8 byte).

Figura 9-5 Funzione CDBL

Esempio

```
10 A = 454.67
20 PRINT A;CDBL(A)
RUN
 454.67  454.67001342773
Ok
```

CINT (PROGRAMMA/IMMEDIATO)

Converte un qualsiasi argomento di tipo numerico in un intero arrotondando la parte frazionaria (se la parte frazionaria è ⩾.5 l'intero è arrotondato per eccesso, in caso contrario per difetto).

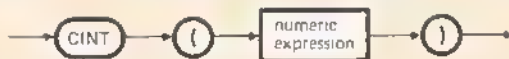



Figura 9-6 Funzione CINT

Esempio

```
PRINT CINT(45.67)
 46
Ok
```

Note

Se l'argomento non è compreso tra i valori -32768 e 32767 si verifica un errore di "Overflow".

Vedere anche le funzioni FIX e INT, che forniscono anch'esse valori interi.

COS (PROGRAMMA/IMMEDIATO)

Calcola il coseno dell'argomento.

Figura 9-7 Funzione COS

Esempio

```
10 X = 2*COS(.4)
20 PRINT X
RUN
 1.84212
Ok
```

Note

L'argomento fornito alla funzione viene assunto come la misura di un angolo espressa in radianti.

La valutazione di COS è effettuata in singola precisione.

CSNG (PROGRAMMA/IMMEDIATO)

Converte un argomento di tipo numerico in un numero in singola precisione (4 byte).

Figura 9-8 Funzione CSNG

Esempio

```
10 A# = 975.3421#
20 PRINT A#; CSNG(A#)
RUN
 975.3421  975.342
Ok
```

Note

Si vedano anche le funzioni CINT e CDBL per la conversione di numeri nei formati interi e a doppia precisione.

EXP (PROGRAMMA/IMMEDIATO)

Eleva la costante "e" (dove "e" vale 2.71828) ad una potenza pari al valore dell'argomento.

Figura 9-9 Funzione EXP

Esempio

```
10 X = 5
20 PRINT EXP (X-1)
RUN
 54.5982
Ok
```

Note

Il valore dell'argomento deve essere ≤ 88.7228 altrimenti viene visualizzato il messaggio di "Overflow"; viene fornito come risultato il numero massimo che la macchina può rappresentare con il segno appropriato e l'esecuzione continua.

La valutazione di EXP viene effettuata in singola precisione.

FIX (PROGRAMMA/IMMEDIATO)

Calcola la parte intera dell'argomento (troncamento).

Figura 9-10 Funzione FIX

Esempi

```
PRINT FIX(58.75)
 58
Ok

PRINT FIX(-58.75)
-58
Ok
```

Note

FIX(X) è equivalente a SGN(X)*INT(ABS(X)).

A differenza di INT, FIX nel caso di argomenti negativi non dà il valore immediatamente inferiore (vedere il secondo esempio).

FRE (PROGRAMMA/IMMEDIATO)

Calcola lo spazio di memoria non utilizzato dal programma BASIC.

Figura 9-11 Funzione FRE

ELEMENTO DI SINTASSI	SIGNIFICATO
dummy argument	è una espressione numerica o stringa. Il valore che la funzione calcola è lo stesso qualunque sia l'argomento dato.

Esempi

```
PRINT FRE(Ø)
 14542
Ok
```

```
PRINT FRE(X$)
 14542
Ok
```

FRE ("") prima di calcolare il numero di byte disponibili forza un "garbage collection" (eliminazione degli spazi vuoti). Inoltre il BASIC fa automaticamente questa operazione nel caso che si abbia saturazione della memoria utente.

INT (PROGRAMMA/IMMEDIATO)

Dà l'intero più grande inferiore o uguale all'argomento.

Figura 9-12 Funzione INT

Esempi

```
PRINT INT(99.89)
 99
Ok

PRINT INT(-12.11)
-13
Ok
```

Note

Si noti la differenza tra INT e FIX. In presenza di valori negativi per INT il risultato è sempre inferiore o uguale all'argomento, mentre per FIX è sempre maggiore o uguale.

LOG (PROGRAMMA/IMMEDIATO)

Calcola il logaritmo naturale di un argomento positivo.

Figura 9-13 Funzione LOG

Dove:

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	deve essere positiva. Altrimenti si verifica il messaggio di errore "Illegal function call"

Esempio

```
PRINT LOG(45/7)
 1.86Ø75
Ok
```

Note

Poichè $\log_a x = \dfrac{\log_e x}{\log_e a}$, il logaritmo in base 1Ø (o in qualsiasi altra base) può essere facilmente calcolato per mezzo della funzione LOG.

Se fosse necessario calcolare il logaritmo in base 1Ø di più valori nell'ambito di un programma, sarebbe allora opportuno definire una funzione utente che consenta detto calcolo.

Ad esempio è sufficiente scrivere in testa al programma:

```
1Ø DEF FNLOG1Ø(X)=LOG(X)/LOG(1Ø)
```

e richiamare la funzione FNLOG1Ø ove necessario, passando a questa l'argomento desiderato.

La valutazione della funzione LOG viene fatta in semplice precisione.

Ritorna un numero casuale compreso tra Ø ed 1. La stessa sequenza di numeri casuali viene generata ogni volta che il programma viene eseguito, a meno che il generatore dei numeri casuali venga riinizializzato (vedere l'istruzione RANDOMIZE).

Figura 9-14 Funzione RND

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	< Ø ricomincia la stessa sequenza di numeri casuali. =Ø ripete l'ultimo numero generato > Ø (oppure omesso, cioè RND) viene generato il prossimo numero casuale della sequenza

Esempio

```
1Ø FOR I=1 TO 5
2Ø PRINT INT(RND*1ØØ);
3Ø NEXT
RUN
 8  25  77  68  7
Ok
```

Note

Sebbene il numero venga chiamato casuale, esso viene in realtà estratto da un ciclo prefissato di numeri (circa un milione in tutto).

Dal momento che il ciclo inizia per ogni esecuzione, lo stesso programma fornisce lo stesso risultato ogni volta che viene eseguito.

Quando tutti i numeri sono stati utilizzati, il ciclo inizia nuovamente.

Per modificare la sequenza dei numeri casuali ogni volta che il programma viene eseguito, si deve usare l'istruzione RANDOMIZE.

RANDOMIZE (PROGRAMMA/IMMEDIATO)

Modifica il generatore di numeri casuali.

Figura 9-15 Istruzione RANDOMIZE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	il suo valore deve essere compreso tra -32768 e 32767. Se il valore non è un intero, viene arrotondato all'intero più prossimo. Questo numero viene utilizzato per definire il punto iniziale di una nuova sequenza di numeri casuali. Se invece viene omesso, l'esecuzione del programma viene sospesa ed il BASIC richiede un valore visualizzando: Random Number Seed (-32768 to 32767)? prima di eseguire l'istruzione RANDOMIZE

Note

Se il generatore dei numeri casuali non è riinizializzato, la funzione RND dà la stessa sequenza di numeri casuali ogni volta che il programma viene eseguito. Per modificare la sequenza dei numeri casuali per ogni esecuzione del programma, si ponga l'istruzione RANDOMIZE all'inizio del programma e si modifichi l'argomento ad ogni esecuzione.

E' anche possibile generare numeri casuali compresi in un dato intervallo. Per generare la sequenza tra A e B si usi la formula

```
FIX((B+1-A)*RND+A)
```

Esempi

```
10 RANDOMIZE
20 FOR I=1 TO 5
30 PRINT RND;
40 NEXT I
RUN
Random Number Seed (-32768 TO 32767)? 3 (l'utente digita 3 CR)
 .88598  .484668  .586328  .119426  .709225
Ok
RUN
Random Number Seed (-32768 to 32767)? 4 (l'utente digita 4 CR per una
nuova sequenza)
 .803506  .162462  .929364  .292443  .322921
Ok
RUN
Random Number Seed (-32768 to 32767)? 3 (stessa sequenza della prima
esecuzione)
 .88598  .484668  .586328  .119426  .709225
Ok
```

SGN (PROGRAMMA/IMMEDIATO)

Dà 1 se l'argomento è positivo, Ø se è uguale a zero e -1 se è negativo.

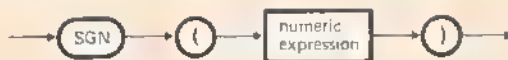



Figura 9-16 Funzione SGN

Esempio

ON SGN(X)+2 GOTO 100,200,300

salta a:

- 100 se X < 0
- 200 se X = 0
- 300 se X > 0

SIN (PROGRAMMA/IMMEDIATO)

Calcola il seno dell'argomento

Figura 9-17 Funzione SIN

Esempio

```
PRINT SIN(1.5)
 .997495
Ok
```

Note

Si assume che il valore dell'argomento sia quello di un angolo misurato in radianti.

SIN viene valutato in singola precisione.

SQR (PROGRAMMA/IMMEDIATO)

Calcola la radice quadrata dell'argomento.

Figura 9-18 Funzione SQR

Esempio

```
10 FOR X = 10 TO 25 STEP 5
20 PRINT X, SQR(X)
30 NEXT
RUN
 10            3.16228
 15            3.87298
 20            4.47214
 25            5
Ok
```

Note

L'argomento deve essere maggiore o uguale a zero, altrimenti si ha il messaggio "Illegal function call". SQR viene valutata in singola precisione.

TAN (PROGRAMMA/IMMEDIATO)

Calcola la tangente dell'argomento

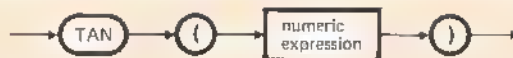



Figura 9-19 Funzione TAN

Esempio

```
10 Y = Q*TAN(X)/2
```

Note

Si assume che il valore dell'argomento sia quello di un angolo misurato in radianti.

Se nell'esecuzione di TAN si verifica un "overflow", viene visualizzato il messaggio di "Overflow", viene fornito come risultato il numero massimo che la macchina può rappresentare con il segno appropriato e l'esecuzione continua.

TAN viene valutata in singola precisione.

FUNZIONI STRINGA DI SISTEMA

Sono funzioni intrinseche che calcolano un valore stringa o un valore numerico e consentono l'utilizzazione di uno o più argomenti numerici e/o stringa.

Semplificano operazioni stringa quali l'estrazione di una sottostringa da una stringa.

Tutte le funzioni stringa di sistema sono elencate qui di seguito in ordine alfabetico.

Nota

In questo paragrafo descriveremo anche l'istruzione MID$, perchè strettamente connessa alla funzione MID$.

ASC (PROGRAMMA/IMMEDIATO)

Calcola un valore numerico che è il codice decimale ASCII del primo carattere di una stringa data.

Figura 9-20 Funzione ASC

Esempio

```
10 X$ = "TEST"
20 PRINT ASC(X$)
RUN
 84
Ok
```

Note

Se il valore dell'argomento è la stringa nulla, si verifica un errore (Illegal function call).

Vedere la funzione CHR$ per la conversione da codice decimale ASCII a carattere.

CHR$ (PROGRAMMA/IMMEDIATO)

Calcola una stringa di un carattere il cui codice decimale ASCII è il valore dell'argomento.

Figura 9-21 Funzione CHR$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	l'espressione numerica viene valutata ed arrotondata all'intero più prossimo. Deve collocarsi nell'ambito di Ø e 255 e viene interpretata come un codice decimale ASCII. Se non cade in questo intervallo si ha il messaggio "Illegal function call"

Esempio

```
PRINT CHR$(66)
B
Ok
```

Note

La funzione CHR$ viene spesso usata per inviare un carattere speciale al terminale. Ad esempio, può essere lanciato il carattere ASCII "BEL" (CHR$ (7)) quale prefazione ad un messaggio di errore o la tabulazione di pagina (CHR$(12)) per cancellare il video e riportare il cursore alla posizione iniziale.

Si veda la funzione ASC per la conversione da carattere ASCII al corrispondente codice decimale.

HEX$ (PROGRAMMA/IMMEDIATO)

Converte un numero decimale in una stringa esadecimale.

Figura 9-22 Funzione HEX$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	l'espressione numerica è arrotondata all'intero più prossimo prima che la funzione HEX$ venga valutata

Esempio

```
10 INPUT X
20 A$ = HEX$(X)
30 PRINT X "DECIMAL IS " A$ " HEXADECIMAL"
RUN
? 32
 32 DECIMAL IS 20 HEXADECIMAL
Ok
```

Note

Si veda la funzione OCT$ per la conversione ottale.

INKEY$ (PROGRAMMA/IMMEDIATO)

Ritorna una stringa di un carattere: questo è il primo carattere impostato in tastiera oppure è la stringa nulla se non vi sono caratteri impostati in attesa di essere elaborati. I caratteri non vengono visualizzati e vengono inoltrati al programma, eccetto per CTRL C che interrompe l'esecuzione del programma.

Figura 9-23 Funzione INKEY$

Esempio

VIDEO	COMMENTI
1000 'Timed Input Subroutine 1010 RESPONSE$="" 1020 FOR I%=1 TO TIMELIMIT% 1030 A$=INKEY$:IF LEN(A$)=0 THEN 1060 1040 IF ASC(A$)=13 THEN TIMEOUT%=0:RETURN 1050 RESPONSE$=RESPONSE$+A$ 1060 NEXT I% 1070 TIMEOUT%=1:RETURN	Questa routine ritorna due valori: - RESPONSE$ che contiene la stringa impostata - TIMEOUT% che equivale a 0 se l'utente imposta una stringa di caratteri prima di completare un dato numero di cicli FOR/NEXT (numero pari a TIMELIMTT%), altrimenti equivale a 1 Nota: la funzione LEN viene descritta più avanti in questo capitolo.

INPUT$ (PROGRAMMA/IMMEDIATO)

Ritorna una stringa di una lunghezza specificata, impostata da tastiera o letta da un file su disco. I caratteri non sono visualizzati e sono inoltrati al programma eccetto per **CTRL** **C** che interrompe l'esecuzione del programma.

Figura 9-24 Funzione INPUT$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
length	espressione numerica arrotondata all'intero più prossimo. Specifica la lunghezza della stringa
file number	è il numero di buffer associato al file (v. Cap. 12)

Esempi

VIDEO	COMMENTI
1Ø OPEN"I",1,"DATA" 2Ø IF EOF (1) THEN 5Ø 3Ø PRINT HEX$(ASC(INPUT$(1,#1))); 4Ø GOTO 2Ø 5Ø PRINT 6Ø END	Questo programma produce un listing di un file sequenziale in caratteri esadecimali Nota: EOF=-1 quando si raggiunge la fine del file (v. Capitolo 12)
11Ø X$=INPUT$(1) 12Ø IF X$="S" THEN END	Impostare S per arrestare l'esecuzione e qualsiasi altro carattere per continuare

INSTR (PROGRAMMA/IMMEDIATO)

Ricerca la prima occorrenza di una sottostringa in una stringa e dà la posizione in cui essa viene individuata.

Figura 9-25 Funzione INSTR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
start position	è una espressione numerica arrotondata all'intero più prossimo che specifica dove la ricerca deve iniziare. Il suo valore deve essere compreso tra 1 e 255. Qualora venga omesso si assume il valore 1
string	è una espressione stringa il cui valore è la stringa da scandire
substring	è una costante o una variabile stringa di cui si deve cercare la prima occorrenza

Esempio

VIDEO	COMMENTI
1Ø X$ = "ABCDEB" 2Ø Y$ = "B" 3Ø PRINT INSTR(X$,Y$);INSTR(4,X$,Y$) RUN 2 6 Ok	Si noti che la posizione in cui viene trovata la prima occorrenza della sottostringa viene sempre calcolata dall'inizio della stringa originaria, anche nell'eventualità che venga indicata una posizione di partenza (start position)

Valori Speciali

SE...	ALLORA...
start position >LEN (string)	il valore della funzione è Ø
start position non è compreso tra i valori 1 e 255	si ha il messaggio di errore: "Illegal function call"
string è una stringa nulla ("")	il valore della funzione è Ø

substring non è individuata	il valore della funzione è Ø
substring è una stringa nulla e start position è specificata	il valore della funzione è uguale a quello di start position
substring è una stringa nulla e la start position non è specificata	il valore della funzione è 1

LEFT$ (PROGRAMMA/IMMEDIATO)

Fornisce una sottostringa estraendo i caratteri più a sinistra di una stringa data per una lunghezza pari a quella assegnata.

Figura 9-26 Funzione LEFT$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string	è una espressione stringa il cui valore corrisponde alla stringa dalla quale la sottostringa deve essere estratta
length	è una espressione numerica arrotondata all'intero più prossimo il cui valore (da Ø a 255) rappresenta la lunghezza della stringa ritornata

Esempio

```
10 A$ = "BASIC LANGUAGE"
20 B$ = LEFT$(A$,5)
```

```
30 PRINT B$
RUN
BASIC
Ok
```

Note

SE...	ALLORA...
length=0	il valore della funzione è una stringa nulla
length non è compreso tra 0 e 255	si ha il messaggio di errore "Illegal function call" (Richiamo di funzione illegale)
length>=LEN(string)	il valore della funzione è l'intera stringa

LEN (PROGRAMMA/IMMEDIATO)

Calcola la lunghezza di una stringa specificata.

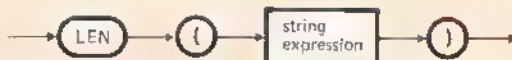



Figura 9-27 Funzione LEN

Esempio

```
10 X$ ="PORTLAND, OREGON"
20 PRINT  LEN(X$)
RUN
 16
Ok
```

Note

La funzione LEN conta tutti i caratteri (stampabili o meno) e anche gli spazi.

Estrae una sottostringa da una stringa, iniziando da una data posizione di carattere. Si può specificare la lunghezza della sottostringa richiesta. In caso contrario verranno ritornati tutti i caratteri della stringa da quella posizione alla fine.

Figura 9-28 Funzione MID$

Dove

ELEMENTO DI SINTASSI	COMMENTI
string	è una espressione stringa il cui valore corrisponde alla stringa dalla quale la sottostringa deve essere estratta
start position	è una espressione numerica arrotondata all'intero più prossimo il cui valore (>=1 e <= della lunghezza della stringa) specifica la posizione del carattere d'inizio della sottostringa estratta
length	è una espressione numerica arrotondata all'intero più prossimo il cui valore (da Ø a 255) rappresenta la lunghezza della sottostringa da estrarre. Se essa viene omessa, vengono estratti tutti i caratteri dalla posizione definita da "start position" alla fine della stringa. Se length = Ø la funzione fornisce la stringa nulla.

Esempio

```
LIST
10 A$="GOOD "
20 B$="MORNING EVENING AFTERNOON"
30 PRINT A$;MID$(B$,9,7)
Ok
RUN
GOOD EVENING
Ok
```

Note

SE...	ALLORA...
start position > LEN(string)	il valore della funzione è una stringa nulla
start position=0	si ha il messaggio di errore "Illegal Function call in line nnnnn" (Argomento illegale nel numero di linea)
length è omessa OPPURE il numero dei caratteri è inferiore a quello specificato da length	vengono estratti tutti i caratteri a partire da "start position" fino alla fine della stringa

MID$ - Istruzione (PROGRAMMA/IMMEDIATO)

Sostituisce in tutto o in parte una data stringa con un'altra stringa.

Figura 9-29 Istruzione MID$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string	è una variabile stringa il cui valore rappresenta la stringa di cui bisogna sostituire una parte
start position	è un'espressione numerica il cui valore, arrotondato all'intero più vicino deve essere >=1 e <= alla lunghezza della stringa data. Questo valore specifica la posizione di carattere dove deve iniziare la sostituzione.
length	è un'espressione numerica il cui valore viene arrotondato all'intero più vicino. Esso deve essere compreso tra Ø e 255 e rappresenta la lunghezza della stringa risultante. Se il parametro length è omesso, vengono sostituiti tutti i caratteri a partire da start position fino alla fine di replacing string. Tuttavia, indipendentemente dal fatto che length venga o meno specificato, la sostituzione dei caratteri non va mai oltre alla lunghezza della stringa originaria.
replacing string	è un'espressione stringa il cui valore sostituisce i caratteri della stringa originaria a partire da start position

Esempio

```
1Ø A$= "KANSAS CITY, MO"
2Ø MID$(A$,14)= "KS"
3Ø PRINT A$
RUN
KANSAS CITY, KS
Ok
```

Note

SE...	ALLORA...
start position > LEN (string)	MID$ ritorna la stringa nulla
start position = Ø	si ha il messaggio d'errore: "Illegal function call"
length è omessa	vengono sostituiti tutti i caratteri a partire da start position fino alla fine di replacing string.
length =Ø	MID$ ritorna la stringa nulla
si cerca di sostituire caratteri oltre alla lunghezza della stringa originaria	la sostituzione termina dopo l'ultimo carattere della stringa originaria

OCT$ (PROGRAMMA/IMMEDIATO)

Calcola una stringa che rappresenta il valore ottale di un argomento

Figura 9-30 Funzione OCT$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	l'espressione numerica è arrotondata all'intero più prossimo prima che la funzione OCT$ venga valutata

Esempio

```
PRINT OCT$(24)
 30
Ok
```

Note

Vedere la funzione HEX$ per le conversioni esadecimali

RIGHT$ (PROGRAMMA/IMMEDIATO)

Fornisce una sottostringa estraendo i caratteri più a destra per una lunghezza pari a quella assegnata.

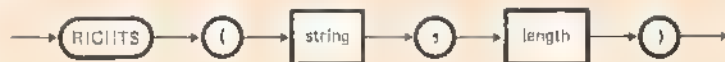



Figura 9-31 Funzione RIGHT$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string	è un'espressione stringa il cui valore corrisponde alla stringa originale dalla quale la sottostringa deve essere estratta

length	è una espressione numerica arrotondata all'intero più vicino, il cui valore compreso tra Ø e 255 rappresenta la lunghezza della stringa richiesta

Esempio

```
1Ø A$="DISK BASIC"
2Ø PRINT RIGHT$(A$,5)
RUN
BASIC
Ok
```

Note

SE...	ALLORA...
length=Ø	il valore della funzione è la stringa nulla
length>=LEN(string)	il valore della funzione è la stringa data

SPACE$ (PROGRAMMA/IMMEDIATO)

Fornisce una stringa di spazi della lunghezza richiesta.

Figura 9-32 Funzione SPACE$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	viene arrotondata al valore intero più vicino. Questo valore deve essere compreso tra Ø e 255 per evitare l'errore di "Illegal function call". Indica il numero di spazi cioè la lunghezza della stringa richiesta

Esempio

```
1Ø FOR I=1 TO 5
2Ø X$=SPACE$(I)
3Ø PRINT X$;I
4Ø NEXT I
RUN
  1
   2
    3
     4
      5
Ok
```

Note

Vedasi anche la funzione SPC nel prossimo paragrafo.

STR$ (PROGRAMMA/IMMEDIATO)

Converte il valore di un'espressione numerica in forma di stringa.

Figura 9-33 Funzione STR$

Esempi

VIDEO	COMMENTI
5 REM ARITHMETIC FOR KIDS 10 INPUT "TYPE A NUMBER";N 20 ON LEN (STR$(N)) GOSUB 30,100,200,300,400,500 : :	Il numero introdotto da tastiera viene assegnato alla variabile N. Questo valore viene convertito in un valore stringa tramite la funzione STR$.
LIST 10 A$ = STR$ (70) 20 PRINT A$ Ok RUN 70 Ok	70 (argomento di STR$) è un numero, ma il contenuto della variabile A$ è una stringa di due caratteri, il cui valore è ancora 70 (da intendersi però come il carattere ASCII 7 seguito dal carattere ASCII 0)
LIST 10 A!=1.3 20 A# =VAL(STR$(A!)) 30 PRINT A# Ok RUN 1.3 Ok	La conversione alla linea 20 fa sì che il valore di A! venga memorizzato in A# senza perdere precisione

Note

VAL esegue la funzione inversa (vedere VAL)

STRING$ (PROGRAMMA/IMMEDIATO)

Crea una stringa di una lunghezza specificata, i cui caratteri sono tutti uguali o al carattere il cui codice ASCII è specificato, o al primo carattere di una stringa specificata.

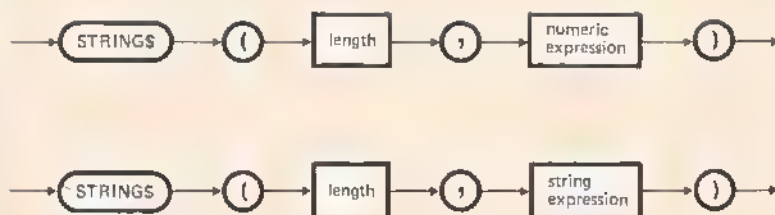



Figura 9-34 Funzione STRING$

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
length	è una espressione numerica arrotondata all'intero più prossimo. Specifica la lunghezza (da Ø a 255) della stringa risultante
numeric expression	viene arrotondata all'intero più prossimo. Specifica il codice decimale ASCII (da Ø a 255) il cui carattere corrispondente viene utilizzato per formare la stringa richiesta
string expression	viene valutata. Il suo primo carattere viene utilizzato per formare la stringa richiesta

Esempio

```
1Ø X$=STRING$(1Ø,45)
2Ø PRINT X$"MONTHLY REPORT"X$
RUN
----------MONTHLY REPORT----------
Ok
```

VAL (PROGRAMMA/IMMEDIATO)

Converte la rappresentazione stringa di un numero nel suo valore numerico.

Figura 9-35 Funzione VAL

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string expression	l'espressione stringa viene valutata. Quando vi sono spazi iniziali o finali, caratteri di tabulazione o di interlinea questi vengono eliminati. La stringa risultante, se è una rappresentazione numerica valida, viene convertita in un numero. Se la stringa comprende un carattere non numerico il risultato della funzione è zero. Ad esempio: VAL (" -3") dà -3 VAL ("ABC") dà Ø

Esempio

```
1Ø READ NAME$,CITY$,STATE$,ZIP$
2Ø IF VAL(ZIP$)<9ØØØØ OR VAL(ZIP$)>96699 THEN
   PRINT NAME$ TAB(25) "OUT OF STATE"
3Ø IF VAL (ZIP$)>=9Ø8Ø1 AND VAL(ZIP$)<=9Ø815 THEN
   PRINT NAME$ TAB(25) "LONG BEACH"
```

Note

STR$ esegue la funzione inversa (vedere STR$).

FUNZIONI DI INPUT/OUTPUT E FUNZIONI SPECIALI DI SISTEMA

Queste funzioni facilitano l'esecuzione di operazioni di input/output, le conversioni di valori, la gestione degli errori, il posizionamento del cursore, le allocazioni delle aree di memoria, ecc. Tali funzioni sono elencate qui di seguito.

Note

In questo paragrafo includiamo anche le parole stringa riservate DATE$ e TIME$ (che possono essere usate o come funzioni, se appaiono in una espressione, o come variabili, se scritte alla sinistra del segno di uguale in una istruzione di assegnazione).

DATE$/TIME$ (PROGRAMMA/IMMEDIATO)

Sono elementi del PCOS che possono essere letti o inizializzati da BASIC per mezzo di questi nomi stringa riservati.

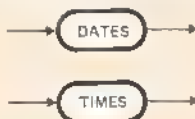



Figura 9-36 DATE$ e TIME$

Note

La data e l'ora possono essere impostate sia in PCOS per mezzo del comando SSYS o in BASIC per mezzo di una istruzione di assegnazione. La data può essere impostata sia come mm:dd:yy, o come mm:dd:yyyy. L'ora viene impostata come hh:mm:ss. L'utente può usare propri delimitatori. (Qualsiasi carattere ASCII tranne le cifre). Per più dettagliate informazioni vedere "Professional Computer Operating System (PCOS) - Guida Utente".

Esempio

VIDEO	COMMENTI
100 IF DATE$="04:30:82" THEN 3000 . . .	l'istruzione 100 controlla che la data sia quella desiderata
500 DATE$="05/06/1981" . . .	l'istruzione 500 imposta la data (cambiando anche il delimitatore)
600 PRINT TIME$ . . .	l'istruzione 600 visualizza l'ora
700 TIME$="07:40:15"	l'istruzione 700 imposta l'ora

CVD (PROGRAMMA/IMMEDIATO)

Converte una stringa di 8 caratteri in un numero in doppia precisione.

Vedere Capitolo 12.

CVI (PROGRAMMA/IMMEDIATO)

Converte una stringa di 2 caratteri in un intero;

Vedere Capitolo 12.

CVS (PROGRAMMA/IMMEDIATO)

Converte una stringa di 4 caratteri in un numero in singola precisione.

Vedere Capitolo 12.

EOF (PROGRAMMA)

Dà -1 se viene raggiunta la fine del file.

Vedere Capitolo 12.

ERL (PROGRAMMA/IMMEDIATO)

Dà il numero di linea della linea in cui è stato trovato l'errore.

Vedere Capitolo 13.

ERR (PROGRAMMA/IMMEDIATO)

Dà il numero del codice di errore.

Vedere Capitolo 13.

LOC (PROGRAMMA/IMMEDIATO)

Dà il numero del record appena letto o scritto (file random), o il numero di settori letti o scritti da quando il file è stato aperto (file sequenziali).

Vedere Capitolo 12.

LPOS (PROGRAMMA/IMMEDIATO)

Ritorna la posizione attuale della testina della stampante (nell'ambito del buffer di linea della stampante).

Figura 9-37 Funzione LPOS

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
dummy argument	è una qualsiasi espressione numerica o stringa. Il risultato non dipende dall'argomento dato

Esempio

```
100 IF LPOS(X)>60 THEN LPRINT CHR$(13)
```

MKD$ (PROGRAMMA/IMMEDIATO)

Converte un numero in doppia precisione in una stringa di 8 caratteri.

Vedere Capitolo 12.

MKI$ (PROGRAMMA/IMMEDIATO)

Converte un numero intero in una stringa di 2 caratteri.

Vedere Capitolo 12.

MKS$ (PROGRAMMA/IMMEDIATO)

Converte un numero in singola precisione in una stringa di 4 caratteri.

Vedere Capitolo 12.

SPC (PROGRAMMA/IMMEDIATO)

Inserisce gli spazi nelle istruzioni PRINT o LPRINT.

Figura 9-38 Funzione SPC

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	viene arrotondata all'intero più vicino. Indica il numero di spazi da inserire nell'immagine visualizzata sia tra due dati sia all'inizio o alla fine dell'immagine. Il numero degli spazi deve essere compreso tra i valori Ø e 255 (onde evitare l'errore "Illegal function call")

Esempio

```
PRINT "OVER" SPC(15) "THERE"
OVER               THERE
Ok
```

Note

In una istruzione PRINT o LPRINT la funzione SPC deve essere seguita o da un punto e virgola o da uno spazio.

Vedere anche la funzione SPACE$.

TAB (PROGRAMMA/IMMEDIATO)

In una istruzione di PRINT o di LPRINT, la funzione TAB posiziona il cursore o la testina della stampante alla posizione specificata.

Figura 9-39 Funzione TAB

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	viene arrotondata all'intero più vicino. Il valore dell'espressione deve essere compreso tra 1 e 255 (onde evitare l'errore "Illegal function call"). Il valore minimo è 1, l'ampiezza della linea -1 è il limite destro. Questo valore specifica la posizione del cursore (o della testina della stampante) in una linea.

Esempio

```
10 PRINT "NAME" TAB(25) "AMOUNT":PRINT
20 READ A$,B$
30 PRINT A$ TAB(25) B$
40 DATA "G.T.JONES","$25.00"
RUN
NAME                    AMOUNT

G.T.JONES               $25.00
Ok
```

Note

Se la posizione del cursore o della testina scrivente della stampante è oltre la posizione indicata dal valore dell'argomento, TAB fa si che il cursore o la testina si posizionino nella giusta posizione sulla linea successiva.

VARPTR (PROGRAMMA/IMMEDIATO)

Formato 1. Fornisce l'indirizzo di memoria del primo byte del dato associato alla variabile specificata.

Formato 2. Nel caso di file sequenziali fornisce l'indirizzo iniziale del buffer di I/O associato al file. Nel caso di file random fornisce l'indirizzo iniziale del buffer di I/O associato al file, definito tramite l'istruzione FIELD.

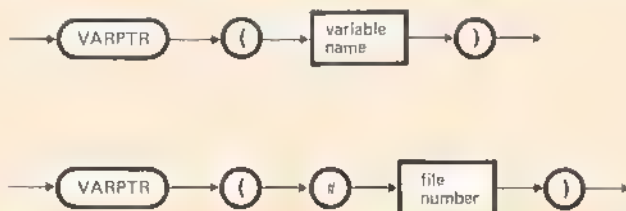



Figura 9-40 Funzione VARPTR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable name	qualsiasi tipo di variabile (numerica, stringa o matrice). L'indirizzo ritornato sarà un'intero compreso tra -32768 e 32767. Nota: Se l'indirizzo ottenuto è negativo, si deve aggiungere 65536 per ottenere l'indirizzo effettivo
file number	numero del buffer associato al file

Esempio

```
100 X%=VARPTR(A(Ø))
```

Note

Se, prima dell'esecuzione della VARPTR, non è stato assegnato alcun valore alla variabile (e questa è una variabile semplice) si verificherà l'errore "Illegal funzione call".

La VARPTR viene generalmente usata per ottenere l'indirizzo di una variabile o di una matrice affinchè possa essere passata ad un sottoprogramma in linguaggio Assembler. La funzione viene usata nella forma VARPTR (A(Ø)) per ottenere l'indirizzo del primo elemento della matrice.

10. SOTTOPROGRAMMI

SOMMARIO

Spesso, una stessa sequenza di istruzioni deve essere eseguita più di una volta nell'ambito di uno stesso programma. In questo caso l'utente non deve riscrivere questa sequenza più volte, ma può scrivere un sottoprogramma che può essere richiamato in qualsiasi punto del programma. Alla fine dell'esecuzione del sottoprogramma, il controllo ritorna all'istruzione successiva a quella del richiamo.

L'M20 consente di utilizzare due tipi diversi di sottoprogrammi richiamabili da un programma BASIC:

- sottoprogrammi scritti in BASIC ("BASIC Subroutines")
- sottoprogrammi scritti in linguaggio ASSEMBLER M20 o comandi PCOS.

Il seguente capitolo illustra questi due tipi di sottoprogrammi e i meccanismi di richiamo.

INDICE

BASIC SUBROUTINES

Una subroutine BASIC è costituita da un insieme qualsiasi di istruzioni BASIC ed è parte integrante del programma. Di norma (ma non necessariamente) inizia con un'istruzione REM e termina con un'istruzione RETURN. E' buona norma di programmazione scrivere le subroutine una dopo l'altra alla fine del programma e chiudere il programma principale (prima dell'inizio della prima subroutine) con un'istruzione END, oppure GOTO, oppure STOP.

Una subroutine può essere richiamata da un'istruzione GOSUB oppure ON...GOSUB. Alla fine dell'esecuzione della subroutine, il controllo ritorna all'istruzione successiva a quella del richiamo.

Diremo che una subroutine è "pendente" se il controllo non è stato ancora restituito al programma principale quando si è verificata una interruzione. Ogni eventuale modifica al programma residente in memoria (cancellazione, modifica di linee ecc...) impedirà di ridare il controllo alla subroutine.

Il seguente esempio illustra il meccanismo di richiamo di una subroutine (istruzioni GOSUB e RETURN).

PROGRAMMA

```
 1Ø REM Programma
        Principale
  .
  .
  .
 5Ø GOSUB 25Ø
 6Ø PRINT X
  .
  .
  .
24Ø GOTO 5ØØ
25Ø REM Sub1
26Ø Z=SQR(T)
  .
  .
  .
29Ø RETURN
  .
  .
  .
5ØØ END
```

COMMENTO

Quando viene eseguita l'istruzione 5Ø del programma principale (GOSUB) il controllo viene trasferito all'istruzione 25Ø (che è la prima istruzione della subroutine).

La subroutine viene quindi eseguita e quando si incontra l'istruzione 29Ø (RETURN), il controllo viene trasferito all'istruzione 6Ø, cioè alla prima istruzione dopo la GOSUB.

L'istruzione 24Ø (GOTO), evita una eventuale attivazione non corretta della subroutine

Se un programma richiama una stessa subroutine più di una volta, alla fine dell'esecuzione della subroutine il controllo viene sempre trasferito alla istruzione successiva all'ultima GOSUB (o ON...GOSUB) che è stata eseguita.

Per esempio, consideriamo un programma che contenga le seguenti istruzioni:

PROGRAMMA	COMMENTI
1Ø REM Programma Principale . . . 5Ø GOSUB 25Ø 6Ø PRINT X . . . 14Ø GOSUB 25Ø 15Ø IF X > 32 THEN 3Ø . . . 24Ø GOTO 5ØØ 25Ø REM Sub1 26Ø Z=SQR(T) . . . 29Ø RETURN . . . 5ØØ END	Quando la subroutine viene attivata tramite l'istruzione 5Ø (GOSUB), il controllo viene trasferito, alla fine dell'esecuzione della subroutine, all'istruzione 6Ø. Quando la subroutine viene di nuovo attivata tramite l'istruzione 14Ø (GOSUB), il controllo viene trasferito, alla fine dell'esecuzione della subroutine, all'istruzione 15Ø

Una subroutine può essere anche richiamata da un'altra subroutine. In questo caso, diremo che la subroutine richiamata è "nested" (annidata) nella subroutine che la ha attivata.

Questo meccanismo può essere ripetuto più volte: il numero di subroutine "nested" contemporaneamente attive, (cioè con istruzione RETURN non ancora eseguita), è limitato solo dalla memoria disponibile.

Ogni istruzione GOSUB, sia essa nel programma principale o in una subroutine, viene sempre associata ad una istruzione RETURN. Questa istruzione RETURN è quella che trasferisce il controllo all'istruzione successiva alla GOSUB. Questo tipo di associazione viene fatto in modo dinamico (cioè durante l'esecuzione): la prima istruzione RETURN eseguita viene associata all'ultima GOSUB eseguita, la seconda RETURN alla penultima GOSUB, e così via.

PROGRAMMA	COMMENTO
10 REM Programma Principale . . . 800 GOSUB 1500 810 . . . 1490 END 1500 REM Sub1 . . . 1900 GOSUB 2500 1910 . . . 2490 RETURN 2500 REM Sub2 . . . 3000 RETURN	- 800 GOSUB 1500 trasferisce il controllo alla subroutine Sub1 - 1500 REM Sub1 individua l'inizio della subroutine Sub1 - 1900 GOSUB 2500 trasferisce il controllo da Sub1 a Sub2 ("nested" subroutine) - 2500 REM Sub2 individua l'inizio della subroutine Sub2 - 3000 RETURN trasferisce il controllo all'istruzione successiva all'ultima GOSUB eseguita (cioè all'istruzione 1910) - 2490 RETURN trasferisce il controllo alla istruzione successiva alla penultima GOSUB eseguita (cioè all'istruzione 810)

GOSUB/RETURN (PROGRAMMA)

GOSUB richiama una subroutine passando il controllo al primo numero di linea della subroutine.

RETURN trasferisce il controllo alla prima istruzione successiva all'ultima GOSUB (oppure ON...GOSUB) eseguita.

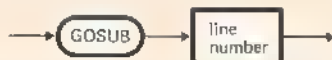



Figura 10-1 Istruzione GOSUB

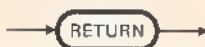



Figura 10-2 Istruzione RETURN

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
line number	è il primo numero di linea di una subroutine

Caratteristiche

UNA SUBROUTINE PUO'...	COMMENTI
iniziare con qualsiasi istruzione (ad esclusione di NEXT e WEND)	per esempio una subroutine può iniziare con REM, LET, FOR,...ecc.. E' buona norma di programmazione iniziare sempre una subroutine con REM (o con un'istruzione che termina con un campo commento)

finire con un'istruzione RETURN	è buona norma di programmazione finire sempre una subroutine con un'istruzione RETURN. In ogni caso RETURN deve essere l'istruzione eseguita per ultima, dato che è l'unica istruzione che consente di ridare il controllo al programma principale. Una subroutine può avere anche più di una istruzione RETURN (ad esempio quando è strutturata in modo da avere più diramazioni ognuna delle quali richieda di ridare il controllo al programma principale)
essere richiamata in qualunque punto del programma, e un numero qualsiasi di volte	se un programma richiama una stessa subroutine più di una volta, il controllo viene trasferito dalla subroutine all'istruzione che segue la GOSUB (oppure la ON...GOSUB) eseguita per ultima
essere inserita in qualsiasi punto del programma	però è buona norma di programmazione scrivere le subroutine una dopo l'altra alla fine del programma, e chiudere il programma (prima dell'inizio della prima subroutine) con un'istruzione END, oppure GOTO, oppure STOP
richiamare un'altra subroutine	il numero di subroutine "nested" contemporaneamente attive, cioè con istruzione RETURN non ancora eseguita), è limitato solo dalla memoria disponibile
accedere ad ogni variabile di programma	tutte le variabili definite nel programma "principale" sono note anche alla subroutine. Queste possono quindi operare su ogni variabile senza restrizioni, (modificandone anche il valore)

Esempi

VIDEO	COMMENTI
LIST 10 DEFINT A-Z 'definisce tutte le variabili intere 20 INPUT "Introduci 3 interi"; A,B,C 30 LET X=A 40 LET Y=B 50 GOSUB 110 60 LET X=M 70 LET Y=C 80 GOSUB 110 90 PRINT "MCD di";A;B;C"=";M 100 GOTO 190 110 LET Q=INT(X/Y) 'subroutine per trovare l'MCD di X e Y 120 LET R=X-Q*Y 130 IF R=0 THEN 170 140 LET X=Y 150 LET Y=R 160 GOTO 110 170 LET M=Y 180 RETURN 190 END Ok RUN Introduci 3 interi? 1377,2916,405 L'MCD di 1377 2916 405 = 81 OK RUN Introduci 3 interi? 4,3333,67 L'MCD di 4 3333 67 = 1 Ok	questo è un programma atto ad evidenziare l'uso di una subroutine. La subroutine utilizza l'algoritmo di Euclide, per trovare il massimo comune divisore (MCD) di tre numeri interi (A, B e C). Questi vengono introdotti da tastiera: i primi due A e B vengono assegnati rispettivamente alle variabili X e Y (vedi istruzioni 30 e 40) e la subroutine determina il loro MCD (vedi istruzioni da 110 a 180). L'MCD trovato viene assegnato alla variabile X nell'istruzione 60 e il terzo numero (C) viene assegnato alla variabile Y nell'istruzione 70. La subroutine viene richiamata di nuovo (vedi istruzione 80) per trovare l'MCD di questi due numeri. Il risultato è l'MCD dei tre numeri. Essi vengono visualizzati insieme con il loro MCD tramite l'istruzione 90. Nota: L'istruzione 10 definisce tutte le variabili intere, dato che il programma opera solo su numeri interi
LIST 10 INPUT "Introduci N>0";N% 20 IF N%<=0 THEN 10 30 GOSUB 50 40 END 50 REM SUB1 (Somma di Interi) 60 S%=(N%*(N%+1))/2	Questo programma calcola la somma dei numeri interi da 1 a N (dove N è introdotto da tastiera) e su richiesta, la somma dei quadrati di questi numeri. Il programma ha due subroutine SUB1

```
70 PRINT "Somma di Interi da 1
        a ";N%;"=";S%
80 INPUT "Somma di Quadrati
        (S/N)";X$
90 IF X$="S" THEN GOSUB 110
100 RETURN
110 REM SUB2 (Somma di Quadrati)
120 S2%=(N%*(N%+1)*(2*N%+1))/6
130 PRINT "Somma di Quadrati da
         1 a ";N%;"=";S2%
140 RETURN
Ok
RUN
Introduci N>0? 5
Somma di Interi da 1 a  5= 15
Somma di Quadrati (S/N)? S
Somma di Quadrati da 1 a  5= 55
Ok
```

e SUB2 inserite alla fine (istruzione da 50 a 100 e da 110 a 140).

Vengono prima eseguite le istruzioni 10, 20 e 30. L'istruzione 30 (GOSUB) richiama la subroutine SUB1 e le istruzioni di quest'ultima vengono eseguite in sequenza fino all'istruzione 90.

Questa esegue un test:

- se il valore di X$ (introdotto da tastiera) è diverso da "S", il controllo passa all'istruzione 100 (RETURN) e quindi all'istruzione 40 (END)

- se il valore di X$ è uguale a "S", viene richiamata la subroutine SUB2 (che è quindi "nested"). Quando si arriva alla istruzione 140 (RETURN di SUB2), il controllo passa all'istruzione 100 (RETURN di SUB1) e quindi alla 40 (END).

ON...GOSUB/RETURN (PROGRAMMA)

L'istruzione ON...GOSUB richiama una subroutine scelta tra n subroutine specificate.

L'istruzione RETURN trasferisce il controllo alla prima istruzione successiva all'ultima ON...GOSUB (o GOSUB) eseguita.

Figura 10-3 Istruzione ON...GOSUB

Figura 10-4 Istruzione RETURN

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	il suo valore specifica la subroutine da richiamare: - Se il suo valore è 1, viene richiamata la subroutine il cui primo numero di linea è il primo della lista. - Se il suo valore è 2, viene richiamata la subroutine il cui primo numero di linea è il secondo della lista, e così via. - Se il suo valore non è intero, viene arrotondato all'intero più vicino. - Se il suo valore è uguale a Ø, o è maggiore del numero di line number della lista (ma minore o uguale a 255), viene eseguita l'istruzione successiva alla ON...GOSUB.

	- Se il suo valore è negativo o maggiore di 255, si ha il seguente messaggio d'errore: Illegal funtion call
line number	ogni line number specificato deve essere il primo numero di linea di una subroutine

Esempio

VIDEO	COMMENTI
LIST 1Ø INPUT "Introduci 1,2 o 3";K% 2Ø ON K% GOSUB 4Ø,5Ø,6Ø 3Ø END 4Ø PRINT "SUB1":RETURN 5Ø PRINT "SUB2":RETURN 6Ø PRINT "SUB3":RETURN Ok RUN Introduci 1,2 o 3? 2 SUB2 Ok	se l'utente introduce il numero 1, il programma visualizza SUB1, se introduce 2 visualizza SUB2, se introduce 3 visualizza SUB3. In ognuno di questi casi una istruzione RETURN trasferisce il controllo alla END; Se l'utente introduce un intero tra Ø e 255, diverso da 1, 2 o 3 il programma non visualizza nulla.

COMANDI PCOS RICHIAMABILI DA BASIC E SOTTOPROGRAMMI ASSEMBLER

Le istruzioni CALL ed EXEC consentono di richiamare da BASIC sottoprogrammi Assembler o comandi PCOS.

Entrambe queste istruzioni hanno la stessa funzione ma:

- EXEC viene usata quando gli argomenti da passare ai corrispondenti parametri sono valori costanti;

- CALL viene usata quando gli argomenti da passare ai corrispondenti parametri sono o costanti o variabili di programma.

Le istruzioni CALL ed EXEC possono essere utilizzate sia in un programma BASIC sia in modo immediato, ma sono più spesso usate in un programma.

Alla fine dell'esecuzione di un sottoprogramma Assembler o di un comando PCOS, il controllo ritorna all'istruzione seguente a quella del richiamo (se CALL o EXEC sono state usate in un programma) oppure ritorna al BASIC Stato Comandi (se CALL o EXEC sono state usate in modo immediato).

L'utilizzo di CALL o EXEC consente di eseguire un programma che interagisce con il PCOS, per esempio per settare i valori delle variabili globali di sistema prima di eseguire altri programmi BASIC o comandi PCOS.

Alla fine dell'esecuzione di questo programma è possibile restare in BASIC o passare in PCOS (tramite il comando SYSTEM).

Normalmente un tale programma di inizializzazione viene denominato INIT.BAS.

Questo è un nome di file riservato. L'M20 dopo aver caricato il PCOS e il BASIC ricerca questo file su entrambi i drive. Se viene trovato, l'M20 entra in BASIC ed esegue INIT.BAS.

Note

Alla fine dell'esecuzione di una istruzione CALL o EXEC che attiva un comando SBASIC del PCOS i nuovi valori settati da SBASIC non verranno presi in considerazione per il programma attuale (perchè potrebbe essere distrutto), ma diventeranno operativi per i programmi successivi, quando il sistema rientra in BASIC.

Solo l'istruzione EXEC (non la CALL) consente di selezionare da BASIC le unità di input/output (per ulteriori dettagli vedasi "Professional Computer Operating System (PCOS)- Guida Utente").

CALL (PROGRAMMA/IMMEDIATO)

Richiama da BASIC un comando PCOS o un sottoprogramma ASSEMBLER, passando al sottoprogramma variabili di programma o costanti.

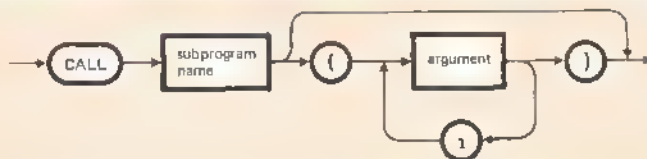



Figura 10-5 Istruzione CALL

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
subprogram name	può essere o il nome di un comando PCOS o il nome di un sottoprogramma ASSEMBLER. Può essere una costante stringa o una variabile stringa
argument	può essere una costante o una variabile semplice o un'espressione il cui valore viene trasferito al parametro corrispondente (il meccanismo di passaggio argomenti/parametri è del tutto analogo a quello visto per le funzioni - vedi Capitolo 9). Se "argument" è un argomento di output (cioè se il sottoprogramma ritorna un valore al BASIC), il nome dell'argomento deve essere preceduto dal simbolo @. Un argomento variabile (di input o di output) deve essere inizializzato prima di eseguire la CALL. Se l'argomento è numerico deve essere intero.

Esempi

VIDEO	COMMENTI
1Ø DEFINT A-C . . . 3Ø FILE$="VOL1:FILEØØ1" 4Ø SIZE%=1Ø 5Ø CALL "fn"(FILE$,SIZE%) . . .	l'istruzione 5Ø richiama il comando PCOS FNEW, passando l'identificatore di file tramite la variabile stringa FILE$ e la dimensione del file tramite la variabile numerica SIZE%. L'istruzione 1ØØ richiama il coman-

```
90 C$ ="LIST"
100 CALL "pK"(&41,C$)
 .
 .
 .
220 A = 10
230 B = 20
240 C = 200
250 CALL "SUB121"(A,B,@C)
 .
 .
 .
```

do PCOS pkey specificando il tasto tramite la costante esadecimale &41 (cioè A, vedere tabella ASCII, appendice A) e la stringa corrispondente tramite la variabile C$.

L'istruzione 250 richiama il sottoprogramma ASSEMBLER SUB121 passando due argomenti di input (A e B) e uno di output (@C). Si noti che questi argomenti sono stati preventivamente inizializzati.

Note

Il comando LTERM del PCOS viene di solito richiamato da BASIC tramite l'istruzione CALL.

Esso ritorna un valore intero (0, 1, 2) corrispondente all'ultimo tasto di chiusura impostazione utilizzato (↵, S1, S2).

Il comando PCOS CI (Communication Interface) viene di solito richiamato da BASIC per inviare o ricevere caratteri in linea tramite la porta RS-232-C.

Altri comandi PCOS (LABEL, SPRINT, BVOLUME, ecc.) sono di solito richiamati da BASIC.

Per ulteriori informazioni vedere "Professional Computer Operating System (PCOS) - Guida Utente" e per il comando CI vedere "I/O con Periferiche Esterne - Guida Utente".

EXEC (PROGRAMMA/IMMEDIATO)

Richiama da BASIC un comando PCOS o un sottoprogramma ASSEMBLER, passando valori costanti al sottoprogramma.

Figura 10-6 Istruzione EXEC

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string expression	il suo valore viene interpretato come il nome di un sottoprogramma seguito da una lista di argomenti costanti

Note

Se EXEC richiama un comando PCOS, il contenuto della espressione stringa dopo EXEC deve coincidere con il comando così come l'utente avrebbe dovuto impostarlo per eseguirlo in PCOS.

Se l'istruzione EXEC richiama un sottoprogramma in linguaggio Assembler, il contenuto dell'espressione stringa dopo EXEC è una lista di parametri separati da virgole; il primo specifica il nome del sottoprogramma ed i successivi specificano gli argomenti da passare al sottoprogramma.

Nota: Gli argomenti non sono racchiusi in parentesi e possono solo essere costanti.

Esempi

VIDEO	COMMENTI
. . . 100 EXEC "pK '#', 'RUN V1:CASHFLOW'"	l'istruzione 100 permette di richiamare da BASIC il comando PCOS PKEY, per assegnare la stringa "RUN V1:CASHFLOW" al tasto # .
. . . 150 EXEC "fp 1:MY.FILE/SECRET"	Si noti che le stringhe: - # - RUN V1:CASHFLOW
. . . 180 A$="FN 1:FILEA,15"	devono essere racchiuse tra apici (') come se si operasse in PCOS.

```
.
.
.
230 EXEC A$
```

L'istruzione 150 permette di richiamare da BASIC il comando PCOS FPASS per assegnare la password SECRET al file MY.FILE, residente sul disco inserito nell'unità 1.

L'istruzione 230 permette di richiamare il comando PCOS FNEW, specificando il comando tramite il contenuto di una variabile stringa inizializzata con l'istruzione 180.

SYSTEM (PROGRAMMA/IMMEDIATO)

Consente di passare da BASIC a PCOS e di chiudere tutti i file dati.

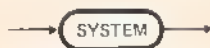



Figura 10-7 Comando SYSTEM

Note

Il comando SYSTEM permette all'utente di ritornare in PCOS. Può essere usato sia in modo immediato sia all'interno di un programma BASIC. Viene frequentemente usato come ultima istruzione di un programma di inizializzazione che utilizza istruzioni CALL ed EXEC per eseguire in sequenza una serie di comandi PCOS e/o sottoprogrammi ASSEMBLER. Per maggiori dettagli vedere "PCOS (Professional Computer Operating System) - Guide dell'Utente".

TASTI PROGRAMMABILI

Usando i tasti **CTRL** e **COMMAND**, assieme a qualsiasi altro tasto (che non sia un tasto shift), l'utente può assegnare un significato particolare a ogni tasto.

Questo può essere un comando PCOS o BASIC, un'espressione, una costante o qualsiasi sequenza di caratteri da impostare frequentemente.

L'assegnazione può essere fatta sia in un programma BASIC per mezzo della CALL "pkey" (o EXEC "pkey") o in PCOS per mezzo del comando PKEY.

A seconda delle esigenze dell'utente, l'assegnazione di una determinata funzione ad un tasto può essere permanente (cioè questa funzione può venir attribuita in modo automatico ad ogni inizializzazione del sistema usando un PCOS "personale") oppure temporanea (cioè valida finché il sistema non viene reinizializzato). Per ulteriori dettagli vedere "PCOS (Professional Computer Operating System) - Guida Utente".

TASTIERE CON VERBI BASIC

Per definire i tasti con i verbi BASIC (siglati sulla tastiera USA-ASCII + verbi BASIC e sulla tastiera Inglese) è disponibile un programma di nome BKEYBOARD.BAS.

Questo programma è costituito da una serie di istruzioni PKEY che definiscono i tasti per generare i verbi BASIC, premendo contemporaneamente il tasto COMMAND.

Il programma risiede sul dischetto standard di sistema.

L'utente può eseguirlo impostando:

bk CR

in PCOS, oppure:

RUN "BKEYBOARD.BAS" CR

in BASIC.

Eseguendo questo programma le definizioni dei tasti restano valide fino alla prossima inizializzazione.

Se l'utente vuole che le definizioni restino valide ad ogni inizializzazione senza bisogno di eseguire ogni volta il programma, deve creare un PCOS personale con il comando PSAVE e inizializzare il sistema con quel PCOS.

Per ulterioni dettagli vedere "PCOS - Guida Utente".

SELEZIONE DA BASIC DELLE UNITA' DI INPUT/OUTPUT

E' possibile da BASIC eseguire un comando PCOS selezionando contemporaneamente una unità di I/O (tramite un'istruzione EXEC o CALL). La selezione permane finché non viene eseguito un altro comando atto a modificarla o non si esca dall'ambiente BASIC. Se però la EXEC (o la CALL) fa una selezione permanente, questa continua ad essere valida anche ritornando in PCOS.

Esempi

Se l'utente imposta...	ALLORA...
ba CR . . . EXEC "VL 1:,+D1:OUT" CR . . . EXEC "-D1:OUT" CR	il sistema carica in memoria l'interprete BASIC ed entra in BASIC (Stato Comandi). La prima istruzione EXEC provoca la visualizzazione della directory del dischetto inserito nel drive 1 e la registrazione di quanto appare su video anche sul file di nome OUT. Questo file viene creato, se non esiste sul dischetto specificato. Se esiste lo ricopre. Il file viene disabilitato come unità di output dalla seconda istruzione EXEC.
ba CR . . . EXEC "vl 1:,+dprt;" CR . . . SYSTEM CR	il sistema entra in BASIC. L'istruzione EXEC provoca la visualizzazione e la stampa della directory del dischetto inserito nel drive 1. Il comando SYSTEM fa ritornare in PCOS e disabilita la stampa come unità di output.

ba CR	il sistema entra in BASIC.
. . . EXEC "+prt" CR	L'istruzione EXEC provoca la stampa di quanto viene visualizzato su video (selezione permanente).
. . . SYSTEM CR	Il comando SYSTEM fa ritornare in PCOS, ma non disabilita la stampa di quanto appare su video, dato che la EXEC ha fatto una selezione permanente.

11. SEGMENTAZIONE DEI PROGRAMMI

SOMMARIO

In questo capitolo illustreremo la tecnica di segmentazione dei programmi e le modalità di passaggio dei dati da un programma a un altro.

Esamineremo in particolare l'istruzione CHAIN (con tutte le sue opzioni) e l'istruzione COMMON. Inoltre ricorderemo l'uso dei comandi RUN e LOAD con l'opzione R.

INDICE

QUANDO USARE LA SEGMENTAZIONE DEI PROGRAMMI

La segmentazione dei programmi significa realizzare una successione di programmi semplici invece di un unico programma complesso.

Questi programmi che devono essere eseguiti uno dopo l'altro, risolvono lo stesso problema di un unico programma complesso. Usando questa tecnica, l'utente può eseguire programmi che altrimenti non potrebbero essere caricati nella memoria dell'M20: la segmentazione dei programmi è una tecnica che può essere utile anche in molti altri casi, alcuni dei quali sono indicati nella seguente tabella.

SE...	ALLORA...
un programma non può essere contenuto nella memoria disponibile	è necessario, per poterlo eseguire, suddividerlo in due o più programmi più semplici, da eseguire in tempi successivi
un programma contiene delle parti che vengono eseguite raramente	può essere opportuno codificare queste parti come programmi distinti da richiamare in memoria quando necessario
un programma ha una parte che deve essere sempre residente, mentre altre parti possono risiedere in memoria temporaneamente (tipicamente queste parti sono routine standard che possono essere utilizzate da molti programmi)	può essere opportuno codificare queste parti come programmi distinti: il segmento che deve essere sempre residente ("radice") richiamerà in memoria per l'esecuzione il primo segmento transiente (overlay). Il secondo overlay verrà quindi richiamato in memoria o dal primo overlay o dalla stessa radice. Ognuno di questi overlay verrà ricoperto (tutto o in parte) dal successivo
un programma è decomponibile in parti funzionali distinte	può essere opportuno codificare queste parti come programmi distinti al fine di ridurre i costi di programmazione

TRASFERIMENTO DATI

La segmentazione dei programmi comporterà la necessità di trasferire dati da un programma al successivo. Ciò può essere fatto in vari modi, come indicato dalla seguente tabella.

SE si utilizza...	ALLORA...
CHAIN insieme a una o più istruzioni COMMON	il BASIC crea un'area "common" che non viene ricoperta dal programma concatenato (il programma residente in memoria viene invece ricoperto). L'area "common" contiene tutte le variabili specificate nell'istruzione COMMON; il programma concatenato può accedere a queste variabili.
CHAIN con l'opzione ALL	tutte le variabili definite dal programma residente in memoria, vengono trasferite al programma concatenato.
CHAIN e il programma attuale accede ad uno o più file dati	il trasferimento dei dati può avvenire anche tramite l'utilizzo di questi file. L'istruzione CHAIN non chiude i file dati. L'utilizzo dei file dati è compatibile con: - le istruzioni CHAIN e COMMON - l'istruzione CHAIN con l'opzione ALL - l'istruzione CHAIN con l'opzione MERGE (ed eventualmente DELETE - vedere la spiegazione dell'opzione DELETE più avanti in questo capitolo).
RUN o LOAD con l'opzione R, e il programma attualmente in memoria accede a uno o più file dati	il trasferimento dati avviene tramite l'utilizzo di questi file. I commandi RUN o LOAD con l'opzione R non chiudono i file dati.

CONCATENAMENTO DEI PROGRAMMI

Come abbiamo già visto, la segmentazione dei programmi può essere realizzata tramite:

- le istruzioni CHAIN e COMMON

- i comandi RUN e LOAD

L'istruzione CHAIN, con tutte le sue opzioni, fornisce un mezzo potente e flessibile per la segmentazione dei programmi.

L'istruzione CHAIN può essere utilizzata secondo tre modalità distinte:

- insieme a una o più istruzioni COMMON, in modo da trasferire l'area common al programma concatenato

- con l'opzione MERGE per poter inserire il programma concatenato nel programma residente in memoria (l'opzione DELETE viene spesso usata in questo caso per cancellare una parte del programma a fine esecuzione, consentendo così di eseguire una serie di "overlay")

- con l'opzione ALL per passare tutte le variabili al programma concatenato

CHAIN (PROGRAMMA)

Consente di concatenare il programma specificato a quello attualmente in memoria permettendo il trasferimento delle variabili. CHAIN lascia i file dati aperti e conserva il valore iniziale dell'indice (Ø oppure 1) degli elementi delle matrici.

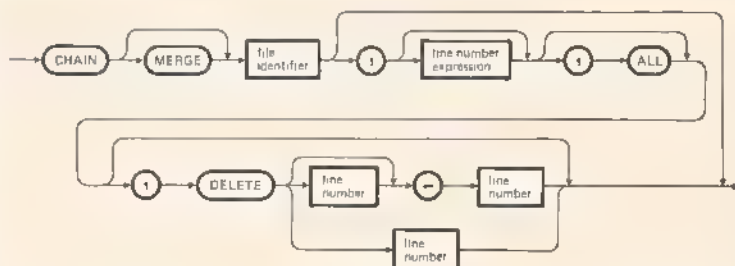



Figura 11-1 Istruzione CHAIN

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
MERGE	viene eseguita un'operazione di MERGE tra il programma attualmente in memoria e il programma concatenato. Quest'ultimo deve essere stato registrato su disco in formato ASCII. Se l'opzione MERGE viene omessa, il programma attualmente in memoria viene ricoperto dal programma concatenato (ad esclusione dell'eventuale area "common"). MERGE viene spesso usata con l'opzione DELETE e line number expression per eseguire una sequenza di "overlay" (v. Esempi). Nota: l'opzione MERGE non modifica i tipi delle variabili. Le variabili definite nel programma concatenante possono quindi essere usate nel programma concatenato. Le funzioni definite dall'utente invece resteranno indefinite dopo che l'operazione di MERGE è stata ultimata.
file identifier	è un'espressione stringa che specifica il programma da concatenare
line number expression	può essere un numero di linea o una espressione il cui valore rappresenta un numero di linea del programma concatenato. Questo parametro individua la linea del programma concatenato che viene eseguita per prima. Questo parametro viene spesso usato insieme con MERGE e DELETE per eseguire una sequenza di overlay. Se questo parametro è omesso, il programma concatenato viene eseguito a partire dall'inizio. Nota: Il comando RENUM non modifica il valore di questo parametro

ALL	se l'istruzione CHAIN comprende l'opzione ALL, i valori di tutte le variabili del programma attualmente in memoria, vengono trasferiti al programma concatenato. Se l'opzione ALL non viene specificata, i dati vengono trasferiti tramite l'area "common" e/o tramite l'uso di file dati.
DELETE	specifica (tramite numeri di linea) che una sezione del programma attualmente in memoria deve essere cancellata. La cancellazione avviene prima che il programma concatenato sia caricato in memoria. DELETE è spesso usato insieme con MERGE e line number expression per eseguire una sequenza di "overlay". Nota: I numeri di linea usati dopo DELETE vengono modificati se si esegue un comando RENUM.

Esempi

VIDEO	COMMENTI
1Ø REM PROG1 2Ø COMMON A1,B1,C1$ . . . 1ØØ CHAIN "PROG2" 11Ø END	il programma PROG1 concatena il programma PROG2 e trasferisce a PROG2 i valori A1,B1,C1$ (tramite l'uso di un'area "common"). PROG2 risiede sul disco inserito nell'ultima unità selezionata.

```
10 REM PROG2
20 COMMON A2$, B2$
 .
 .
 .
80 CHAIN "PROG3",200
90 END
```

il programma PROG2 concatena PROG3 e trasferisce a PROG3 i valori di A2$ e B2$ (tramite l'uso di un'area "common").

Il programma PROG3 viene eseguito a partire dalla linea 200.

PROG3 risiede sul disco inserito nell'ultima unità selezionata.

```
10 REM PROG10
 .
 .
 .
50 CHAIN "1:PROG11", 100, ALL
60 END
```

il programma PROG10 concatena PROG11 e trasferisce a questo tutte le sue variabili.

PROG11 viene eseguito a partire dalla linea 100.

PROG11 risiede sul dischetto inserito nell'unità 1.

```
10 REM ROOT
 .
 .
 .
100 CHAIN MERGE "V1:OVERLAY1", 1000
110 END
```

il programma ROOT concatena con l'opzione MERGE il programma OVERLAY1 (registrato in formato ASCII sul disco V1).

Questo viene eseguito a partire dalla linea 1000

```
1000 REM OVERLAY1
 .
 .
 .
1500 CHAIN MERGE "V1:OVERLAY2",
     1000, DELETE 1000-1500
1510 END
```

il programma OVERLAY1 concatena con l'opzione MERGE il programma OVERLAY2 (registrato in formato ASCII sul disco V1).

Questo viene eseguito a partire dalla linea 1000. Prima del suo caricamento però le istruzioni dalla linea 1000 alla 1500 vengono cancellate

COMMON (PROGRAMMA)

Definisce un'area common che non viene ricoperta dal programma concatenato e permette il trasferimento delle variabili.

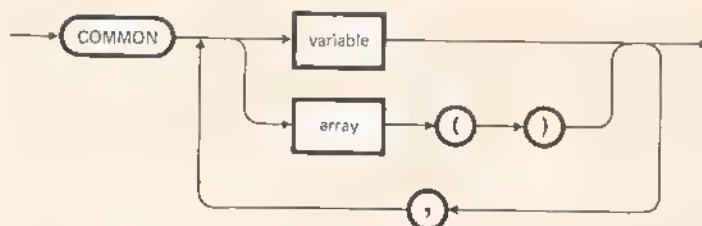



Figura 11-2 Istruzione COMMON

Esempi

VIDEO	COMMENTI
10 REM PG1 20 COMMON A1,B1,C1,D1$ . . . 80 CHAIN "VOL2:PG2" 90 END	Le istruzioni COMMON vengono usate unitamente all'istruzione CHAIN. Un programma può avere una o più istruzione COMMON. le variabili specificate in queste istruzioni vengono allocate nell'area common a partire dall'inizio dell'area e secondo la sequenza di scrittura nel programma.
10 REM PG2 20 PRINT A1,B1,C1,D1$ . . . . 120 END	Il programma concatenato non deve necessariamente specificare, tramite istruzioni COMMON, le variabili COMMON specificate dal programma concatenante. Il programma concatenato farà riferimento a queste variabili usando gli stessi nomi indicati nel programma concatenante.

```
10 REM PG1
20 DEFDBL C
30 COMMON A1,B1,C1,D1$
 .
 .
 .
90 CHAIN "VOL2:PG2"
100 END

10 REM PG2
20 DEFDBL C
 .
 .
 .
130 END
```

Nel nostro esempio i valori delle variabili A1,B1,C1 e D1$ del programma PG1 vengono trasferiti al programma concatenato PG2, che ne visualizza i valori (vedi istruzioni 20).

Eventuali istruzioni di definizione di tipo (DEFINT, DEFSNG, DEFDBL, DEFSTR) relative a variabili common devono precedere le istruzioni COMMON e dovranno apparire anche nel programma concatenato in modo da conservare la coerenza dei tipi delle variabili. Si noti in questo caso la presenza delle due istruzioni DEFDBL nei due programmi PG1 e PG2

```
10 REM PROGRAM1
20 COMMON A$,B$,C$
30 COMMON A$,A1
 .
 .
 .
100 END
```

E'buona norma di programmazione non scrivere uno stesso nome di variabile (in questo caso A$) in istruzioni COMMON diverse nell'ambito dello stesso programma (né più di una volta in una stessa istruzione COMMON).
In ogni caso le definizioni multiple sono equivalenti a una singola.

```
10 REM PG1
20 DIM A1(15,20)
30 COMMON A1(),B1,C1
 .
 .
 .
100 CHAIN "VOL2:PG2"
110 END
10 REM PG2
 .
50 PRINT A1(1,1)
 .
90 END
```

un'istruzione COMMON può specificare anche nomi di matrici. In tal caso questi nomi devono essere seguiti da una coppia di parentesi. Ogni matrice indicata in una istruzione COMMON deve anche essere dimensionata tramite un'istruzione DIM nel programma concatenante (ma non nel programma concatenato, altrimenti si verificherà un errore "Duplicate Definition in line...").

L'istruzione DIM deve precedere l'istruzione COMMON associata.

```
10 REM mod1
20 A=1:B=2
30 COMMON A,B
40 GOTO 60
50 COMMON C
60 CHAIN "mod3"

10 REM mod2
20 A=1:B=2
30 COMMON A
40 GOTO 60
50 COMMON B
60 CHAIN "mod3"

10 REM mod3
20 PRINT A;B
```

l'istruzione COMMON ha un valore dichiarativo efficace anche se non viene eseguita. Quando si esegue "mod1" si ha "Illegal function call in 50" perché la variabile C non è definita.

Eseguendo invece "mod2", viene chiamato il programma "mod3" che visualizza sia la variabile A che la variabile B, anche se la linea 50 di "mod2" è stata "saltata".

Nota

Le variabili common devono essere inizializzate nel programma concatenante.

12. GESTIONE DI UN FILE SU DISCO

SOMMARIO

Questo capitolo descrive i due tipi di file dati disponibili in BASIC:

- i file sequenziali (sequential file)

- i file ad accesso diretto (Random file)

Esamineremo come questi file vengono creati, aperti, chiusi e come sia possibile leggerli e scriverli.

INDICE

FILE SEQUENZIALI E AD ACCESSO DIRETTO

Un file dati viene creato (cioè reso noto al sistema) tramite:

- il comando FNEW del PCOS, che ne stabilisce il nome e l'estensione iniziale,

oppure

- l'istruzione OPEN, che consente di accedere al file da programma.

L'istruzione OPEN attribuisce anche il nome a un file che non sia già stato creato tramite FNEW o un'altra OPEN. In ogni caso OPEN associa un buffer dati al file (utilizzato dal programma per ogni operazione di Input/Output) e specifica il metodo d'accesso.

Useremo il comando FNEW invece dell'istruzione OPEN per creare un file dati molto esteso di cui si conosca sufficientemente bene la dimensione finale. FNEW riserva un certo numero di settori contigui su disco al file. Ciò può rendere più efficienti le operazioni di Input/Output. Inoltre FNEW garantisce che ci sia spazio sufficiente a contenere il file su disco.

Dal punto di vista implementativo esiste un solo tipo di file (byte stream); ogni file può però essere aperto secondo quattro diversi metodi d'accesso (Input, Output, Append e Random). I primi tre permettono un accesso di tipo sequenziale (si parla quindi di file sequenziali), mentre l'ultimo ammette esclusivamente l'accesso random (si parla quindi di file ad accesso diretto). Il metodo d'accesso di un file può essere modificato ogni volta che il file viene riaperto.

La seguente tabella riassume le caratteristiche fondamentali dei file dati, classificando i file (dal punto di vista utente) in due categorie: sequenziali e ad accesso diretto.

TIPO DI FILE	CARATTERISTICHE	METODO D'ACCESSO
Sequenziale (o "Stream-oriented")	un file sequenziale è una sequenza di caratteri ASCII senza alcun criterio di raggruppamento. Il numero di dati letti o registrati in ogni operazione di Input/Output può variare ed è di norma determinato dal numero di variabili specificate nell'istruzione	Input: lettura sequenziale (un dato dopo l'altro) dall'inizio del file Output: registrazione sequenziale dall'inizio del file. I dati presenti precedentemente sul file sono persi Append: registrazione sequenziale dalla fine del file preesistente. I dati presenti nel file non sono persi.
Ad Accesso Diretto (o "Record-oriented")	Un file ad accesso diretto è una sequenza di dati raggruppati in record. Ogni istruzione di Input/Output può leggere o registrare un record alla volta. I record di un file ad accesso diretto hanno tutti la stessa lunghezza e la stessa struttura.	Random: accesso diretto in lettura o registrazione al record specificato.

FILE SEQUENZIALI

I file sequenziali rappresentano il modo più semplice di memorizzare i dati.

Conviene utilizzare file sequenziali quando i dati hanno un formato libero (cioè non si possono raggruppare in record).

I dati vengono scritti su un file sequenziale uno dopo l'altro (in sequenza) e vengono poi letti nello stesso ordine. Si tengano presenti i seguenti punti:

- quando un file sequenziale viene aperto in Output, i dati vengono registrati in sequenza a partire dall'inizio del file. Se il file conteneva in precedenza dei dati, questi vengono persi.

- quando un file sequenziale viene aperto in Append, i dati vengono registrati in sequenza dopo l'ultimo dato sul file.

- per aggiornare un file sequenziale è necessario aprirlo in lettura (Input), leggerne i dati, aggiornarli e riscriverli su un nuovo file sequenziale aperto in scrittura (Output).

- i dati scritti su un file sequenziale includono di norma dei delimitatori che specificano dove ogni dato inizia e finisce.

- per leggere un file sequenziale l'utente deve aprirlo in Input e deve conoscere il formato dei suoi dati; per esempio se sul file sono memorizzati valori numerici separati da spazi o valori numerici e stringa separati da virgola.

- i dati su un file sono sempre registrati come una stringa di caratteri (un byte per ogni carattere del dato). Per esempio il numero:

  351.27

  richiede 6 byte di memoria su disco, esclusi i delimitatori (che possono essere spazi o virgole).

FILE AD ACCESSO DIRETTO

Conviene utilizzare file ad accesso diretto quando è possibile raggruppare i dati in "record", aventi tutti una data lunghezza.

I file ad accesso diretto richiedono un maggior numero di passi di programma che non i file sequenziali, ma presentano vari vantaggi:

- invece di iniziare a leggere o a scrivere un file dall'inizio, l'utente può leggere o scrivere un record qualsiasi del file.

- per aggiornare un file, non è necessario leggere l'intero file, aggiornare i suoi dati e scriverli su un altro file. L'utente può riscrivere o aggiungere un qualsiasi record senza dover accedere a tutti i record precedenti.

- l'apertura di un file ad accesso diretto consente sia di scrivere che di leggere record dal file utilizzando lo stesso "buffer" per entrambe le operazioni.

APERTURA E CHIUSURA DI UN FILE

Per accedere a un file dati da programma, l'utente deve aprirlo tramite un'istruzione OPEN. Questa specifica l'identificatore di un file, il metodo d'accesso, il numero dei file e, limitatamente ai file ad accesso diretto, la lunghezza dei record.

Il massimo numero di file aperti contemporaneamente può essere stabilito tramite il comando SBASIC del PCOS, o può essere assunto per default (in quest'ultimo caso il suo valore è 3). Il numero massimo non può mai superare 15.

Ogni volta che l'utente apre un file, un buffer viene associato al file. Ogni buffer ha un numero compreso tra 1 e 15. Il numero di buffer viene anche detto numero di file; esso viene usato in ogni operazione di I/O per far riferimento al file preceduto dal simbolo # (che è però opzionale in alcune istruzioni). Il buffer è un'area di transito per i dati che devono essere letti o scritti su file.

Per i file ad accesso diretto, l'utente deve stabilire la struttura del buffer (cioè dei record nel file) fissando la lunghezza (in caratteri) di ogni dato all'interno del buffer tramite l'istruzione FIELD.

L'utente per accedere a un dato file con un'istruzione di input/output, farà riferimento al file tramite il numero del file e non tramite il suo identificatore.

Quando l'utente chiude un file, tramite l'istruzione CLOSE, viene cancellata l'associazione, fatta dalla OPEN, tra il file e il buffer, e non è più possibile accedere al file, a meno che non venga aperto di nuovo. Quando l'utente riapre un file, tramite un'altra OPEN, può associare al file lo stesso o un altro buffer.

OPEN (PROGRAMMA/IMMEDIATO)

Apre un file dati consentendo l'accesso da programma.
Se il file non è già stato creato la OPEN crea anche il file (però l'utente non può creare un file aprendolo con metodo d'accesso "I")

Figura 12-1 Istruzione OPEN

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
access mode	è una costante o una variabile stringa il cui valore può essere: - "A",cioè Append : predispone la registrazione sequenziale dei dati dopo l'ultimo dato. Il file è sequenziale. I dati presenti sul file non vengono persi, i nuovi dati vengono aggiunti di seguito a quelli esistenti - "I" cioè Input : predispone la lettura sequenziale dei dati a partire dall'inizio dal file. Il file è sequenziale - "O" cioè Output : predispone la registrazione sequenziale dei dati a partire dall'inizio del file. Il file è sequenziale. Se il file conteneva già dei dati questi vengono persi. - "R" cioè Random : i record potranno essere letti o scritti in qualsiasi ordine. In questo caso il file è ad accesso diretto. N.B. Se un file sequenziale è vuoto (cioè non contiene dati) i metodi di accesso A e O sono equivalenti
file number	è una espressione numerica, il cui valore (arrotondato all'intero più vicino) deve essere compreso tra 1 e 15. Il numero di file specificato rimane associato al file per tutto il tempo in cui il file resta aperto e viene usato per specificare il file in ogni istruzione di I/O
file identifier	è una costante o una variabile stringa e può specificare: - un file nuovo (cioè sconosciuto al sistema). In questo caso il file viene creato (ad esclusione dei file aperti con metodo d'accesso "I"). - un file già esistente. In questo caso il file viene soltanto aperto.

record length	è un'espressione numerica (arrotondata all'intero più vicino) che, se specificata, stabilisce la lunghezza dei record di un file ad accesso diretto. Questo parametro può essere specificato solo per i file ad accesso diretto. Il suo valore di default è 256 byte. Il suo valore massimo è dato dal parametro record size del comando SBASIC del PCOS. I valori ammessi del record size vanno da 1 a 4Ø96 (con un valore di default di 256 bytes).

Esempi

VIDEO	COMMENTI
. . . 5Ø OPEN "A",1,"V1:EXAMPLE"	L'istruzione 5Ø apre il file sequenziale EXAMPLE residente sul disco V1 in Append, e associa al file il buffer numero 1.
. . . 16Ø OPEN "O",2,"V1:TEST"	L'istruzione 16Ø apre il file sequenziale TEST, residente sul disco V1, in Output e associa al file il buffer numero 2.
. . . 27Ø OPEN "R",3,"V2:F1",8Ø	L'istruzione 27Ø apre il file ad accesso diretto F1, residente sul disco V2, associa al file il buffer numero 3 e stabilisce la lunghezza dei record pari a 8Ø byte.
28Ø OPEN "R",4,"V2:F2",2Ø	L'istruzione 28Ø apre il file ad accesso diretto F2, residente sul disco V2, associa al file il buffer numero 4 e stabilisce la lunghezza dei record pari a 2Ø byte.
. . . 49Ø CLOSE 2	L'istruzione 49Ø chiude il file TEST.
5ØØ OPEN "I",5,"V1:TEST"	L'istruzione 5ØØ riapre il file TEST in Input e associa al file il buffer numero 5.
. . . 6ØØ OPEN "R",2,FILE$,RN	L'istruzione 6ØØ apre un file ad accesso diretto il cui identificatore è il contenuto della variabile stringa FILE$. La lunghezza dei suoi record è data dal valore della variabile numerica RN. Il buffer associato è il buffer 2 reso disponibile dopo l'istruzione 49Ø.

Nota

Non è possibile creare un file con un'istruzione OPEN, se viene specificato come metodo di accesso "I". Un eventuale tentativo si risolve in un errore per "File not found"

CLOSE (PROGRAMMA/IMMEDIATO)

Chiude i file dati

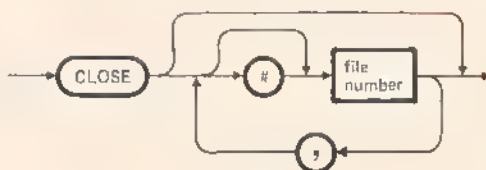



Figura 12-2 Istruzione CLOSE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato rappresenta il numero del buffer associato al file. Questo numero deve essere compreso tra 1 e 15. Un'istruzione CLOSE senza parametri chiude tutti i file che erano stati aperti.

Esempi

VIDEO	COMMENTI
. . . 17Ø CLOSE # 2 . 25Ø A=6 . 29Ø CLOSE 3,5,A . . . 12ØØ CLOSE	L'istruzione 17Ø chiude il file associato al buffer numero 2. L'istruzione 29Ø chiude i file associati ai buffer numero 3,5 e 6 (se A vale 6). L'istruzione 12ØØ chiude tutti i file.

Caratteristiche

SE...	ALLORA...
viene eseguita un'istruzione CLOSE	l'associazione tra il file che è stato chiuso e il suo buffer viene annullata; il buffer può allora essere riutilizzato per aprire qualsiasi file. Un file che è stato chiuso può essere riaperto tramite una istruzione OPEN (nell'ambito dello stesso o di un altro programma). La OPEN può associare al file qualsiasi buffer che in quel momento sia libero
viene eseguita una un'istruzione END o un comando SYSTEM	tutti i file aperti vengono chiusi
l'utente imposta **CTRL RESET**	tutti i file aperti vengono chiusi e tutti i dati nei buffer di I/O, non ancora registrati su disco, vengono persi

viene fatta una qualsiasi modifica al programma (inserimento, modifica di linee etc...)	tutti i file aperti vengono chiusi
viene eseguita una istruzione CHAIN, oppure un comando LOAD (o RUN) con l'opzione R	nessun file viene chiuso
si ha un'interruzione del programma (a seguito di una istruzione STOP, di un messaggio d'errore, dell'impostazione di CTRL C)	nessun file viene chiuso
si cerca di chiudere un file già chiuso o non ancora aperto	l'istruzione non ha effetto

Note

E' sempre conveniente chiudere un file quando non si ha più bisogno di elaborarlo, a meno di dover concatenare un altro programma al programma attuale (tramite CHAIN o RUN con l'opzione R o LOAD con l'opzione R) per accedere allo stesso file con lo stesso metodo di accesso.

Un comando LOAD oppure RUN (senza l'opzione R) oppure un comando SAVE, chiudono tutti i file che sono ancora aperti.

SCRITTURA DI UN FILE SEQUENZIALE

Il file deve essere aperto con metodo d'accesso Output ("O") o Append ("A")

Le istruzioni di Output sono PRINT# , PRINT# USING e WRITE #. PRINT# e WRITE # scrivono i dati in un formato standard, mentre PRINT# USING scrive i dati in un formato definito dall'utente. La differenza tra PRINT # e WRITE # è che:

- PRINT # registra su disco i dati nello stesso formato standard col quale l'istruzione PRINT li visualizza su video.

- WRITE # registra su disco i dati nello stesso formato standard col quale l'istruzione WRITE li visualizza, cioè inserisce una virgola tra un dato e il successivo ed inoltre registra il carattere virgolette davanti e alla fine di ogni valore stringa.

Nota: Può essere usata la funzione LOC per sapere il numero di settori (blocchi di 256 byte) registrati nel file da quando è stato aperto, al fine di evitare il messaggio d'errore "Disk full"

I seguenti passi di programma sono necessari per scrivere dati su un file sequenziale.

PASSO	AZIONE	ESEMPI
1	L'utente deve aprire il file specificando il metodo d'accesso "O" oppure "A"	1Ø OPEN "O",1,"1:F1" . . .
2	L'utente deve scrivere una serie di valori numerici e/o stringa sul file, usando delle istruzioni di Output	. . . 5Ø WRITE# 1,A$,B,C$ . . .
3	L'utente deve ripetere il passo 2 per ogni operazione di Output	. . . 15Ø WRITE #1,A1,B1,C1$ . . . 18Ø WRITE # 1,A2,B2$,C2,D2 . . .

4	Quando non ci sono più dati da registrare, l'utente deve chiudere il file (a meno che non debba essere riutilizzato con lo stesso metodo d'accesso da un altro programma concatenato al precedente)	. . . 300 CLOSE #1 . . .

PRINT # (PROGRAMMA/IMMEDIATO)

Registra dati su un file sequenziale, usando lo stesso formato standard utilizzato da un'istruzione PRINT.

Figura 12-3 Istruzione PRINT#

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file.
expression	può essere un'espressione numerica, di relazione, logica o stringa, il cui valore viene registrato sul file.

Note

Poiché i dati vengono registrati su disco nello stesso formato in cui apparirebbero su video con l'istruzione PRINT, l'utente deve fare attenzione a registrare i dati con gli opportuni delimitatori in modo da poterli poi rileggere in modo corretto.

Caratteristiche

SE...	ALLORA...
viene eseguita un'istruzione PRINT#	i dati vengono registrati in modo sequenziale sul file specificato
il file viene aperto in Output ("O")	il pointer viene posto all'inizio del file, quindi la prima PRINT# registra i dati a partire dall'inizio del file Ad ogni operazione PRINT# ,il pointer avanza, sicché i valori vengono registrati in sequenza
il file viene aperto in Append ("A")	il pointer è posto alla fine, quindi la prima PRINT # registra i dati dopo l'ultimo dato presente nel file. Ad ogni operazione PRINT # il pointer avanza, sicché i valori sono registrati in sequenza
l'utente vuole scrivere correttamente la lista PRINT # per leggere poi i dati registrati tramite una o più istruzioni INPUT#	deve tener presente che un'istruzione PRINT # crea una stringa su disco simile a quella che crea una PRINT sul video PRINT# registra una stringa di dati codificata ASCII. La punteggiatura nella lista PRINT# è molto importante La virgola e il punto e virgola non racchiusi tra virgolette hanno lo stesso effetto di quando sono presenti nelle istruzioni PRINT

l'utente vuole registrare dei valori numerici (che possono essere cioè il risultato di una espressione numerica o di relazione o logica) deve separare le espressioni con una virgola o un punto e virgola.

Se non vuole perdere spazio su disco deve usare il punto e virgola anziché la virgola

Per esempio:

```
LIST
10 OPEN "O", #1,"DATA1"
20 A=1:B=2:C=3
30 PRINT #1,A;B;C
40 CLOSE #1
50 OPEN "I", #1,"DATA1"
60 INPUT #1,A1,B1,C1
70 PRINT A1;B1;C1
80 CLOSE #1
90 END
Ok
RUN
 1  2  3
Ok
30 PRINT #1,A,B,C
RUN
 1  2  3
Ok
```

Se l'utente separa le variabili numeriche A,B e C nell'istruzione 30 con virgole anziché punti e virgola, il risultato è esattamente lo stesso, ma spreca spazio su disco dato che vengono inseriti spazi addizionali tra le variabili

Con il punto e virgola, la stringa su disco sarà:

```
 1  2  3
```

con la virgola, sarà:

```
 1              2              3
```

l'utente vuole registrare valori stringe	deve inserire, in modo esplicito, dei delimitatori se vuole leggere quei valori come stringhe distinte (tremite la INPUT #)
l'utente vuole registrere dei valori stringa che non contengono virgole, punti e virgole, spazi iniziali e finali significativi, ritorno a capo o interlinee	deve usare una virgole, espresse come costente stringe (",") per separare le espressioni stringe nelle PRINT # . In questo modo anche i deti su disco verranno separati con una virgola e potranno quindi essere riletti come stringhe distinte tramite un'istruzione INPUT#

Per esempio:

```
LIST
10 OPEN "O", #1,"DATA1"
20 A$="CAMERA"
30 B$="93605-2"
40 PRINT # 1,A$;B$
50 CLOSE # 1
60 OPEN "I", # 1,"DATA1"
70 INPUT # 1,A1$
80 PRINT A1$
90 CLOSE # 1
100 END
Ok
RUN
CAMERA93605-2
Ok
40 PRINT # 1,A$;",";B$
70 INPUT # 1,A1$,B1$
80 PRINT A1$,B1$
RUN
CAMERA        93605-2
Ok
```

Se l'utente separa A$ e B$ con un punto e virgola nell'istruzione 40, le stringe di ceratteri registrata su disco sarà:

CAMERA93605-2

Poiché non ci sono dei delimitatori, questa stringa non può essere riletta come due stringhe distinte. Per poter fare ciò, bisogne inserire, in modo esplicito, un delimitatore (",") nell'istruzione 40 e correggere in conseguenza enche le istruzioni 70 e 80.

La stringa registreta su disco sarà ellora:

CAMERA,93605-2

Questa stringa verrà letta come due stringhe separate (vedi la seconda esecuzione del programma)

l'utente vuole registrare dei valori stringa che contengono virgole , punti e virgola, spazi iniziali o finali significativi, ritorni a capo o interlinee

deve separare le espressioni stringa con un carattere virgolette ("), espresso come funzione CHR$(34)

Per esempio:

```
LIST
10 OPEN "O",#1,"DATA1"
20 A$="CAMERA, AUTOMATIC"
30 B$="   93605-2"
40 PRINT#1,A$;B$
50 CLOSE#1
60 OPEN "I",#1,"DATA1"
70 INPUT#1,A$,B$
80 PRINT A$;B$
90 CLOSE#1
100 END
Ok
RUN
CAMERAAUTOMATIC   93605-2
Ok
40 PRINT#1,CHR$(34);A$;CHR$(34);CHR$(34);
   B$;CHR$(34)
RUN
CAMERA, AUTOMATIC   93605-2
Ok
```

L'istruzione 40 registra su disco la seguente stringa:

CAMERA, AUTOMATIC   93605-2

e l'istruzione 70 assegna la stringa

CAMERA

alla variabile A$ e la stringa

AUTOMATIC   93605-2

alla variabile B$. Questo si può verificare tramite l'istruzione 80 quando si esegue il pro-

gramma per la prima volta. Se l'utente modifica l'istruzione 4Ø come indicato, la stringa registrata su disco sarà:

"CAMERA, AUTOMATIC""   93605-2"

e l'istruzione 7Ø assegnerà la stringa:

"CAMERA, AUTOMATIC"

alla variabile A$ e la stringa:

"   936Ø5-2"

alla variabile B$. Questo si può verificare con l'istruzione 8Ø, quando si esegue il programma per la seconda volta.

PRINT # USING (PROGRAMMA/IMMEDIATO)

Registra dati su un file sequenziale, usando un formato definito dall'utente allo stesso modo di un'istruzione PRINT USING che però visualizza l'output sul video.

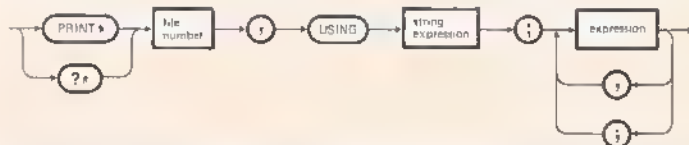



Figura 12-4 Istruzione PRINT # USING

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file
string expression	è la formattazione dei caratteri descritta dettagliatamente nel capitolo 7
expression	può essere un'espressione numerica, di relazione, logica o stringa, il cui valore viene registrato sul file

Note

L'utente deve aver cura di registrare i dati con gli opportuni delimitatori in modo da poterli poi rileggere in modo corretto con un'istruzione INPUT#.

Per esempio, l'istruzione:

```
PRINT# 1,USING"####,##,";A,B,C,D
```

può essere usata per registrare dati numerici senza delimitatori espliciti. La virgola al termine della stringa di formato serve a separare i dati sul file.

Si veda il capitolo 7 per maggiori dettagli sulle prestazioni offerte dall'istruzione PRINT USING

WRITE# (PROGRAMMA/IMMEDIATO)

Registra dati su un file sequenziale, usando lo stesso formato utilizzato dall'istruzione WRITE sul video. Ogni dato viene separato dal precedente tramite una virgola. Le stringhe vengono delimitate dal carattere virgolette ("). Dopo la registrazione dell'ultimo dato della lista, il BASIC introduce un ritorno a capo/interlinea.

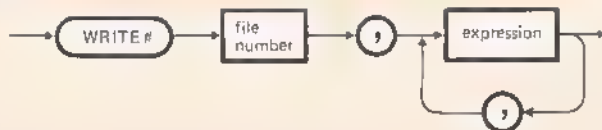



Figura 12-5 Istruzione WRITE #

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file
expression	può essere un'espressione numerica, di relazione, logica o stringa, il cui valore viene registrato sul file

Note

Non è necessario inserire dei delimitatori espliciti nella lista di un'istruzione WRITE #.

Se l'utente vuole registrare una stringa che contiene il carattere virgolette ("), deve usare l'istruzione PRINT invece dell'istruzione WRITE #.

Il carattere virgolette (") può essere inserito all'interno di un valure stringa tramite la funzione CHR$(34), purché il valore stringa non contenga virgole, punti e virgola, spazi iniziali e finali significativi, ritorni a capo e interlinee.

Un carattere virgolette (") può anche appartenere al valore stringa memorizzato in una variabile stringa, se detto valore viene assegnato tramite istruzioni READ e DATA o istruzioni INPUT (LINE INPUT, INPUT #, o LINE INPUT #).

Esempio

VIDEO	COMMENTI
LIST 10 OPEN "O",1,"DATA2" 20 A$="CAMERA" 30 B$="93605-2" 40 WRITE #1,A$,B$ 50 CLOSE #1 60 OPEN "I",1,"DATA2" 70 INPUT#1,A$,B$ 80 WRITE A$,B$ 90 CLOSE #1 100 END OK RUN "CAMERA","93605-2" OK	L'istruzione 40 registra la seguente stringa su disco: "CAMERA","93605-2" L'istruzione 70 assegna "CAMERA" alla variabile A$ e "93605-2" alla variabile B$. Questo si può verificare con l'istruzione 80

LOC (PROGRAMMA/IMMEDIATO)

Nel caso di file sequenziale, LOC ritorna il numero di settori (blocchi di 256 byte) letti da/registrati su file da quando è stato aperto. La funzione LOC può anche essere usata con file ad accesso diretto (vedi seguito).

Figura 12-6 Funzione LOC

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica arrotondata all'intero più vicino. Si tratta del numero del buffer associato al file

Esempio

```
200 IF LOC(2)>30 THEN STOP
```

LETTURA DI UN FILE SEQUENZIALE

Per leggere un file sequenziale l'utente deve aprirlo in Input ("I").

Le istruzioni INPUT# e LINE INPUT# consentono di leggere dati da un file sequenziale.

INPUT# legge uno o più dati, separati da delimitatori, e li assegna ad una lista di variabili numeriche e/o stringa. L'istruzione LINE INPUT# legge un'intera linea (fino al carattere ritorno a capo) e la assegna ad una variabile stringa.

Oltre alle due istruzioni qui indicate, il BASIC consente di utilizzare due funzioni molto utili per il trattamento dei file sequenziali:

- la funzione EOF: consente di verificare se è stata raggiunta la fine di un file e di evitare quindi una ulteriore lettura che darebbe luogo al messaggio d'errore:

  Input past end.

- la funzione LOC : permette di conoscere il numero di settori (blocchi di 256 byte) che sono stati letti dal file, o scritti sul file, da quando è stato aperto.

Per leggere i dati da un file sequenziale, sono richiesti i seguenti passi di programma:

PASSO	AZIONE	ESEMPI
1	L'utente deve aprire il file, specificando "I" come metodo d'accesso.	1Ø OPEN "I", # 2,"DATA" . . .
2	L'utente può quindi leggere una serie di valori numerici e/o stringa dal file, utilizzando le istruzioni INPUT# e/o LINE INPUT#	. . . 5Ø INPUT# 2,X$,Y$,Z . . .
3	L'utente può continuare a leggere i dati, cioè ripetere il passo 2 per un numero arbitrario di volte (fino a raggiungere la fine del file)	. . . 1ØØ INPUT# 2,X1,X2,X3,X4 . . . 15Ø INPUT# 2,U$,W$ . . .
4	Quando ha finito di leggere i suoi dati, l'utente chiuderà il file (a meno che non voglia concatenare un altro programma che debba leggere dati dal file).	. . . 2ØØ CLOSE # 2 . . .

INPUT# (PROGRAMMA/IMMEDIATO)

Legge i dati da un file sequenziale e li assegna a variabili di programma.

Figura 12-7 Istruzione INPUT#

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file
variable	è il nome della variabile che dovrà contenere un dato letto dal file

Note

A differenza dell'istruzione INPUT, l'istruzione INPUT# non visualizza alcun messaggio quando viene eseguita.

Caratteristiche

SE...	ALLORA...
viene eseguita un'istruzione INPUT#	i dati vengono letti dal file in sequenza, cioè, quando il file viene aperto, il "pointer" del file viene posto all'inizio del file. Ogni volta che un dato viene introdotto, il "pointer" si sposta all'inizio del dato successivo. Se l'utente vuole rileggere il file dall'inizio, deve chiudere il file ed aprirlo di nuovo.
l'utente vuole leggere dati senza generare errori	deve conoscere il tipo (numerico o stringa) di ogni dato sul file. I dati sul file dovranno essere separati da opportuni delimitatori (vedi seguito). Dati numerici possono essere introdotti in variabili stringa. Se l'utente introduce un numero in una stringa e ne vuole poi il suo valore numerico effettivo, dovrà usare la funzione VAL ad evitare errori di incompatibilità di tipo.
il BASIC sta per assegnare un dato a una variabile numerica	ignora eventuali spazi iniziali, caratteri di ritorno a capo e interlinea. Quando viene incontrato il primo carattere diverso dai precedenti, il BASIC assume di aver incontrato il primo carattere di un numero. Il numero termina quando viene incontrato uno spazio, oppure un ritorno a capo o una interlinea o una virgola. N.B. Nelle assegnazioni numeriche possono verificarsi delle conversioni se la variabile è di tipo diverso dalla costante (così come può succedere con le istruzioni LET o INPUT o READ nel capitolo 5).

il BASIC sta per assegnare un dato a una variabile stringa	ignora eventuali spazi iniziali, caratteri di ritorno a capo e interlinea. Quando viene incontrato il primo carattere diverso dai precedenti, il BASIC assume di aver incontrato il primo carattere di una stringa.
questo primo carattere è il carattere virgolette (")	la stringa consiste di tutti i caratteri letti fino al successivo carattere virgolette. I caratteri virgolette non fanno parte della stringa (ciò significa che una stringa inclusa tra virgolette su disco, non può comprendere il carattere virgolette)
questo primo carattere <u>non</u> è il carattere virgolette (")	la stringa non è compresa tra virgolette e termina quando viene incontrata una virgola, oppure un ritorno a capo o una interlinea (o dopo che sono stati letti 255 caratteri).

Per esempio, se sul disco sono presenti i seguenti dati:

```
SUBROUTINES, SOTTOPROGRAMMI "COME RICHIAMAR-
LI?"
```

l'istruzione:

```
INPUT # 1,R$,S$,T$
```

assegnerà i valori come segue:

```
R$ = SUBROUTINES
S$ = SOTTOPROGRAMMI "COME RICHIAMARLI ?"
T$ = stringa nulla
```

Se invece sul disco fossero presenti i seguenti dati:

```
SUBROUTINES, SOTTOPROGRAMMI, "COME RICHIAMAR-
LI?"
```

la stessa istruzione assegnerà i seguenti valori:

```
R$ = SUBROUTINES
S$ = SOTTOPROGRAMMI
T$ = COME RICHIAMARLI?
```

LINE INPUT#(PROGRAMMA/IMMEDIATO)

Legge un'intera linea (fino al carattere di ritorno a capo) da un file sequenziale e la assegna ad una variabile stringa

Figura 12-8 Istruzione LINE INPUT#

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore introdotto specifica il numero del buffer associato con il file
string variable	è il nome della variabile a cui sarà assegnata la linea

Caratteristiche

SE...	ALLORA...
viene eseguita un'istruzione LINE INPUT#	viene letta una linea di dati che viene assegnata alla variabile stringa specificata LINE INPUT# legge tutti i caratteri del file fino a incontrare: - un carattere di ritorno a capo, o - un carattere di ritorno a capo/interlinea, o - la fine del file, o

	- il 255mo carattere (questo 255mo carattere è compreso nella stringa)
vengono incontrati caratteri iniziali o altri delimitatori - virgolette, virgole, spazi, e così via	tutti questi caratteri iniziali o tutti questi delimitatori vengono inclusi nella stringa
si vuole leggere dati senza seguire le normali restrizioni riguardanti i caratteri iniziali e terminali	l'utente deve usare istruzioni LINE INPUT#
si vuole leggere come un file dati un programma BASIC in formato ASCII	l'utente deve usare istruzioni LINE INPUT# (è possibile registrare programmi che correggono altri programmi ASCII; rinumerarli, cambiare l'istruzione LPRINT in PRINT, etc...)

Note

LINE INPUT# legge tutti i caratteri nel file fino al carattere di ritorno a capo. Essa quindi salta la sequenza ritorno a capo/interlinea e la successiva LINE INPUT# legge tutti i caratteri fino al prossimo ritorno a capo (se viene incontrata la sequenza interlinea/ritorno a capo, essa viene considerata come facente parte della variabile stringa)

Esempio

VIDEO	COMMENTI
LIST 10 INPUT "PROGRAM IDENTIFIER";P$ 20 OPEN "I",1,P$ 30 K%=0 40 IF EOF(1) THEN 80 50 K%=K%+1 60 LINE INPUT# 1,A$ 70 GOTO 40	Questo programma conta il numero delle linee in un file programma in formato ASCII.

```
80 PRINT P$ " IS" K% "LINES LONG"
90 CLOSE
100 GOTO 10
110 END
Ok
RUN
PROGRAM IDENTIFIER? V1:P1
V1:P1 IS 350 LINES LONG
PROGRAM IDENTIFIER? V1:P2
V1:P2 IS 1520 LINES LONG
PROGRAM IDENTIFIER? ^C
Break in 10
Ok
```

Ogni linea termina con un ritorno a capo/interlinea, così la LINE INPUT# all'istruzione 60 legge un'intera linea alla volta nella variabile fittizia A$. La variabile intera K% conta le linee del programma.

EOF (PROGRAMMA)

Ritorna -1 (cioè vero) se è stata raggiunta la fine di un file sequenziale.
E' opportuno usare EOF per verificare la fine del file mentre lo si legge, per evitare errori di "Input past end"

Figura 12-9 Funzione EOF

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica arrotondata all'intero più vicino. Si tratta del numero del buffer associato al file

Esempio

```
10 DIM A(50)
20 OPEN "I",1,"DATA1"
30 FOR K%=0 TO 50
40 IF EOF(1) THEN 100
50 INPUT#1,A(K%)
60 NEXT K%
    .
    .
    .
```

AGGIORNAMENTO DI UN FILE SEQUENZIALE

Per aggiornare un file sequenziale, l'utente deve leggere l'intero file e quindi registrare i dati aggiornati su un nuovo file di output, come indicato nella tabella che segue:

PASSO	AZIONE
1	L'utente deve aprire il file sequenziale da aggiornare
2	L'utente deve aprire un nuovo file sequenziale
3	L'utente deve leggere una lista di dati e aggiornarli come richiesto
4	L'utente deve registrare i dati aggiornati sul nuovo file
5	L'utente deve ripetere i passi 3 e 4 fino a quando tutti i dati sono stati letti, aggiornati e registrati sul nuovo file; quindi andare al passo 6
6	L'utente deve chiudere entrambi i file (a meno che non debbano essere riutilizzati con lo stesso metodo d'accesso da un altro programma concatenato)

DEFINIZIONE DEL FORMATO DI UN RECORD

Dopo l'apertura di un file ad accesso diretto l'utente deve definire il formato del record tramite un'istruzione FIELD. L'istruzione FIELD organizza il buffer del file ad accesso diretto in modo che diventa possibile passare dati da programma a disco e viceversa.

Il record può essere suddiviso in più campi tramite un'istruzione FIELD, ma il numero totale di byte allocati in un'istruzione FIELD non deve superare la lunghezza del record determinata al momento dell'apertura del file. Diversamente si verifica un errore per "Field overflow". (La lunghezza di default del record è 256 byte).

L'istruzione FIELD stabilisce la dimensione di ciascuno di questi campi e consente ai nomi delle variabili stringa di puntare a ciascun campo. I nomi di questi campi, diversamente dalle stringhe ordinarie che puntano ad un'area in memoria chiamata "spazio stringa", puntano all'area del buffer associato al file.

Tutti i dati, stringa e numeri, devono essere posti nel buffer in formato stringa. Esistono tre paia di funzioni (MKI$/CVI,MKS$/CVS,MKD$/CVD) per convertire i numeri in stringa e viceversa.

Nota: Non è possibile utilizzare il nome di un campo in un'istruzione INPUT, o sul lato sinistro di un'istruzione LET. Quel nome non punterebbe più al campo nel buffer, ma allo spazio stringa; quindi non sarebbe possibile per l'utente accedere a quel campo utilizzando il nome precedentemente assegnatogli.

FIELD (PROGRAMMA/IMMEDIATO)

Definisce i campi nel buffer di un file ad accesso diretto

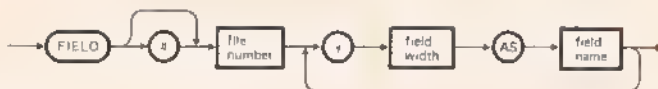



Figura 12-10 Istruzione FIELD

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file
field width	è il numero di byte da allocare nel campo. Un byte corrisponde ad un carattere
field name	è il nome stringa da assegnare al campo definito da "field width"

Esempi

VIDEO	COMMENTI
2Ø FIELD# 1,15 AS NAME$,2Ø AS C$, 1Ø AS P$ . . .	L'istruzione 2Ø alloca le prime 15 posizioni (byte) del buffer# 1 al campo di nome NAME$, le successive 2Ø al campo di nome C$ e le ultime 1Ø al campo di nome P$
8Ø NAME$=B$ (Errato) . . .	Dopo l'esecuzione dell'istruzione 8Ø, NAME$ diventa un nome ordinario di una variabile stringa. L'utente non ha più la possibilità di accedere al primo campo del buffer.
1ØØ LSET NAME$=B$ (Corretto)	L'utente deve invece usare l'istruzione 1ØØ (si vedano di seguito le istruzioni LSET/RSET)

```
30 FIELD# 2,128 AS N1$,128 AS N2$
    .
    .
    .
```

L'utente può usare FIELD tante volte quante vuole per "riorganizzare" il buffer di un file

```
100 FIELD# 2,128 AS N3$,100 AS N4$,
   28 AS N5$
```

La riorganizzazione di un buffer mediante l'istruzione FIELD non cancella i contenuti del buffer; viene cambiato soltanto il modo di accedere al buffer (tramite i nomi dei campi). Così due o più nomi di campi possono fare riferimento alla stessa area del buffer.

```
50 FIELD# 3,16 AS K$(1),112 AS L$(1)
    .
    .
    .
90 FIELD# 3,128 AS DUMMY$,
   16 AS K$(2),112 AS L$(2)
```

In un'istruzione FIELD l'utente può utilizzare una variabile fittizia per "saltare" una parte del buffer e iniziare la strutturazione del buffer da quel punto

Nella seconda istruzione FIELD, DUMMY$ serve a spostare la posizione d'inizio di K$(2) alla posizione 129 del buffer

Nota

E' buona norma che la somma di tutte le dimensioni dei campi sia uguale alla lunghezza del record stabilita dall'istruzione OPEN. In ogni caso questa somma non deve superare la lunghezza del record, diversamente si verifica un errore per "Field overflow".

REGISTRAZIONE DI RECORD SU UN FILE AD ACCESSO DIRETTO

Per registrare record su un file ad accesso diretto, l'utente deve aprire il file, specificando "R" come metodo d'accesso.

L'istruzione PUT-File consente poi la registrazione dei record. Il contenuto del record deve essere stato preparato nell'ambito del buffer ad accesso diretto prima dell'esecuzione dell'istruzione PUT-File, tramite le istruzioni LSET o RSET. Le istruzioni LSET e RSET muovono i dati dalla memoria al buffer del file ad accesso diretto allocando le espressioni stringa nei campi predefiniti.

Se l'espressione stringa utilizza meno byte di quelli che l'utente ha allocato nell'istruzione FIELD, lo spazio allocato in più viene riempito con caratteri blank che possono essere posizionati a sinistra o a destra del valore dell'espressione stringa. L'allineamento a sinistra (v. istruzione LSET) inizia nella prima posizione del campo. L'allineamento a destra (v. istruzione RSET) termina nell'ultima posizione del campo. Quando l'utente deve trasferire valori numerici nel buffer, deve convertirli in stringa tramite le funzioni MKI$, MKS$ e MKD$.

Nota: La funzione LOC ritorna il numero del record appena scritto tramite una PUT-File o letto tramite una GET-File.

Per registrare record su un file ad accesso diretto, sono richiesti i seguenti passi di programma:

PASSO	AZIONE	ESEMPI
1	L'utente deve aprire il file, specificando "R" come metodo d'accesso e (opzionalmente) la lunghezza del record	1Ø DPEN "R",# 1,"1:DIR",22
2	L'utente deve strutturare il buffer tramite una FIELD #	2Ø FIELD# 1,15 AS A$,5 AS B$, 2 AS C$
3	L'utente può quindi inserire i dati nel buffer	1ØØ LSET A$="JOHN JONES" 11Ø LSET B$="U.K." 12Ø LSET C$=MKI$(I%)
4	L'utente può registrare un record (il quinto) sul file	13Ø PUT# 1,5
5	L'utente può registrare un altro record ripartendo dal passo 3. Diversamente deve andare al passo 6	

6	Quando ha finito di registrare i suoi dati l'utente chiuderà il file (a meno che voglia concatenare un altro programma che debba utilizzare lo stesso file con lo stesso metodo d'accesso)	15Ø CLOSE# 1

LSET/RSET (PROGRAMMA/IMMEDIATO)

LSET memorizza un valore stringa allineato a sinistra in un campo del buffer ad accesso diretto, o allinea a sinistra un valore stringa in una variabile stringa.

RSET memorizza un valore stringa allineato a destra in un campo del buffer ad accesso diretto o allinea a destra un valore stringa in una variabile stringa.

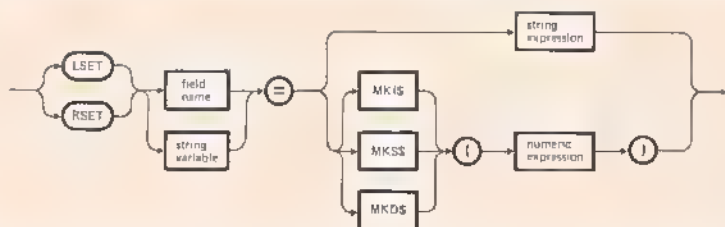



Figura 12-11 Istruzioni LSET/RSET

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
field name	è una variabile stringa che specifica il nome di un campo di un buffer ad accesso diretto
string variable	è il nome di una variabile stringa ordinaria
MKI$/MKS$/MKD$	sono le funzioni che convertono un intero (MKI$), o un valore in singola precisione (MKS$), o in doppia precisione (MKD$), in valore stringa
string expression	è la stringa da allineare a sinistra o a destra in un dato campo
numeric expression	è il valore numerico da convertire in stringa e allineare a sinistra o a destra in un dato campo

Esempi

VIDEO

```
10 OPEN "R", #1,"1:MYFILE/MYPASS",20
20 FIELD #1,10 AS N1$,10 AS N2$
30 LSET N1$="CHARLES"
40 LSET N2$="JAMES"
    .
    .
    .
100 RSET N1$="CHARLES"
110 RSET N2$="JAMES"
    .
    .
    .
200 LSET N1$="CHARLES THOMSON"
    .
    .
    .
```

COMMENTI

Le istruzioni 30 e 40 assegnano i dati sul buffer #1 come segue:

N1$

CHARLES

N2$

JAMES

Le istruzioni 100 e 110 assegnano i dati nel buffer come segue:

N1$

CHARLES

N2$

JAMES

L'istruzione 200 assegna i dati nel buffer come segue:

N1$

CHARLES TH

Nota: Se una stringa è troppo lunga e quindi non può essere contenuta nel campo del buffer specificato, viene troncata sulla destra, indipendentemente dal fatto che sia stata usata l'istruzione LSET o RSET.

```
110 A$=SPACE$(20)
120 RSET A$=N$
```

Le istruzioni LSET e RSET possono anche essere utilizzate con una variabile non associata all'istruzione FIELD per l'allineamento a destra o a sinistra di una stringa in un dato campo. Questa può essere una utile tecnica di formattazione per le operazioni di stampa.

Nell'esempio a sinistra l'istruzione RSET allinea a destra la stringa N$ in un campo di 20 caratteri

MKI$/MKS$/MKD$ (PROGRAMMA/IMMEDIATO)

Queste funzioni cambiano un numero in una stringa.

La funzione MKI$ converte un intero in una stringa di 2 caratteri

La funzione MKS$ converte un valore in singola precisione in una stringa di 4 caratteri

La funzione MKD$ converte un valore in doppia precisione in una stringa di 8 caratteri

Figura 12-12 Funzione MKI$

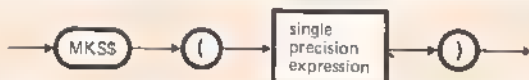



Figura 12-13 Funzione MKS$

Figura 12-14 Funzione MKD$

Esempi

VIDEO	COMMENTI
3Ø LSET D$=MKI$(I%)	Il campo D$ deve ora contenere una rappresentazione di due byte dell'intero I%
1ØØ STR4C$=MKS$(SPV)	Una funzione "make" (MKI$, MKS$, MKD$) viene utilizzata con le istruzioni LSET e RSET dato che queste funzioni utilizzano come argomenti delle stringhe. In questo caso SPV è il nome di una variabile in singola precisione che viene convertita in una stringa di 4 caratteri e assegnata alla variabile stringa STR4C$.

PUT-File (PROGRAMMA/IMMEDIATO)

Registra su un file ad accesso diretto i dati prelevati dal buffer e associato al file.

Figura 12-15 Istruzione PUT-File

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica che specifica il numero del buffer associato al file

record number	è un'espressione numerica che specifica il numero del record nel file. Il numero minimo di record è 1, il numero massimo è 32767. In caso di omissione di questo parametro viene registrato il record corrente.
	Nota: Il record corrente è il record il cui numero è più alto di uno rispetto a quello dell'ultimo record selezionato. La prima volta che l'utente accede a un file ad accesso diretto, il numero del record corrente è definito uguale a 1.

Esempi

VIDEO	COMMENTI
LIST 10 OPEN "r",1,"1:RAND",48 20 FIELD 1,20 AS R1$,20 AS R2$,8 AS R3$ 30 FOR L=1 TO 4 40 INPUT "name";N$ 50 INPUT "address";M$ 60 INPUT "phone";P# 70 LSET R1$=N$ 80 LSET R2$=M$ 90 LSET R3$=MKS$(P #) 100 PUT 1,L 110 NEXT L 120 CLOSE 1 130 END Ok RUN name? super man address? USA phone? 11234621 name? robin hood address? England phone? 23462101 . . . Ok	L'istruzione 10 apre il file ad accesso diretto RAND, con una lunghezza di record di 48 byte, sul disco inserito nell'unità 1. Il numero del file è 1. L'istruzione 20 divide il buffer in campi. L'istruzione 100 registra un record nel file RAND, con il numero di record definibile tramite la variabile di controllo del ciclo iterativo FOR/NEXT

LOC (PROGRAMMA/IMMEDIATO)

Nel caso di file ad accesso diretto, la funzione LOC fornisce il numero del record appena letto tramite un'istruzione GET#, o ritorna il numero del record appena registrato tramite un'istruzione PUT#.

Figura 12-16 Funzione LOC

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica arrotondata all'intero più vicino. E' il numero del buffer associato al file

Esempi

VIDEO	COMMENTI
10 OPEN "R",2,"TOWNS",80 20 FIELD 2,20 AS F1$,20 AS F2$, 20 AS F3$, 20 AS F4$ 30 Y=1 : 100 A$="MILAN " 110 GET 2,Y 120 Y=Y+1 130 IF F1$=A$ THEN PRINT "FOUND IN RECORD";LOC(2):CLOSE:END 140 GOTO 110 :	In questo caso F1$ è il nome di un campo. Se il valore di F1$ è lo stesso di A$, viene visualizzato il numero del record nel quale è stato trovato.

Nota

Se il file è stato aperto, ma non è stata ancora eseguita alcuna operazione di I/O, LOC ritorna il valore Ø.

LETTURA DI RECORD DA UN FILE AD ACCESSO DIRETTO

Per leggere record da un file ad accesso diretto, l'utente deve aprire il file, specificando "R" come metodo d'accesso. L'istruzione GET-File consente la lettura dei record.

GET-File specifica sia il numero del file sia il numero del record da leggere. Con l'esecuzione di una GET-File, i contenuti del record specificato vengono trasferiti nel buffer del file.

Per accedere ad un singolo dato memorizzato nel buffer (tramite il nome del campo) l'utente può usare:

- un'istruzione LET (se vuole assegnarlo a una variabile programma), o

- un'istruzione PRINT, PRINT USING, LPRINT, o LPRINT USING (se vuole visualizzarlo o stamparlo).

Nota: Se l'utente vuole assegnare, visualizzare o stampare il nome di un campo da convertire in un numero deve effettuare la conversione usando la funzione CVI, o CVS o CVD.

Nota: La funzione LOC ritorna il numero del record appena letto tramite una GET-File o scritto tramite una PUT-File.

Per leggere i dati da un file ad accesso diretto, sono richiesti i seguenti passi di programma:

PASSO	AZIONE	ESEMPI
1	L'utente deve aprire il file specificando "R" come metodo d'accesso e (opzionalmente) la lunghezza del record	1Ø OPEN "R",#2,"1:DIR",22
2	L'utente deve strutturare il buffer tramite la FIELD	2Ø FIELD#2,15 AS A$,5 AS B$,2 AS C$

3	L'utente può quindi leggere un record dal file (A indica il numero delrecord)	1ØØ GET#2,A
4	L'utente può estrarre i dati dal buffer o con l'istruzione LET o con l'istruzione PRINT (PRINT USING). I valori numerici (memorizzati in formato stringa all'interno del buffer) devono essere convertiti in numeri usando le funzioni di "conversione": CVI, CVS e CVD	1ØØ A1$=A$ 12Ø PRINT B$ 13Ø I%=CVI(C$)
5	L'utente può continuare a leggere un altro record ripartendo dal passo 3. Diversamente, deve andare al passo 6	
6	Quando ha finito di leggere i suoi dati l'utente chiuderà il file (a meno che voglia concatenare un altro programma che debba leggere dati dallo stesso file con lo stesso metodo d'accesso)	5ØØ CLOSE#2

Nota: In un programma che esegue sia operazioni di input che di output sullo stesso file ad accesso diretto, l'utente spesso può utilizzare una sola istruzione OPEN e una sola istruzione FIELD.

GET-File (PROGRAMMA/IMMEDIATO)

Legge un record da un file ad accesso diretto.

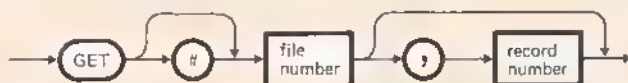



Figura 12-17 Istruzione GET-File

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
file number	è un'espressione numerica il cui valore arrotondato specifica il numero del buffer associato al file
record number	è un'espressione numerica il cui valore arrotondato specifica il numero del record da leggere (cioè da trasferire nel buffer). Se omesso, viene letto il record corrente. Il numero minimo di record è 1, il numero massimo è 32767. Nota: Il record corrente è il record il cui numero supera di uno quello dell'ultimo record selezionato. La prima volta che l'utente accede ad un file ad accesso diretto (senza specificarne il numero nel record) il numero del record corrente è definito uguale a 1.

Esempi

VIDEO	COMMENTI
LIST 10 OPEN "r",1,"1:RAND",48 20 FIELD 1,20 AS R1$,20 AS R2,8 AS R3$ 30 FOR L=1 TO 4 40 GET 1,L 50 PRINT R1$,R2$,CVD(R3$) 60 NEXT 70 CLOSE 80 END Ok RUN Super man USA 11234621 robin hood England 23462101 . : Ok	Questo programma ricerca le informazioni memorizzate nel file specificato. I dati letti nel buffer sono accessibili da programma. Ciò avviene in questo caso tramite l'istruzione PRINT (si veda l'istruzione 50. Si suppone che i dati siano stati registrati in precedenza sul file tramite l'istruzione PUT-File.

CVI/CVS/CVD (PROGRAMMA/IMMEDIATO)

Convertono valori stringa in valori numerici

CVI converte una stringa di 2 caratteri in un intero.

CVS converte una stringa di 4 caratteri in un numero in singola precisione.

CVD converte una stringa di 8 caratteri in un numero in doppia precisione.

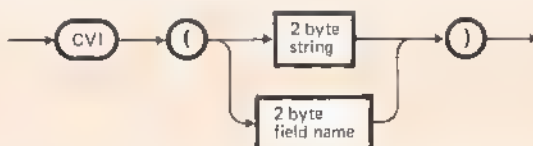



Figura 12-18 Funzione CVI

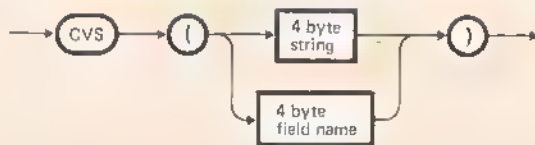



Figura 12-19 Funzione CVS

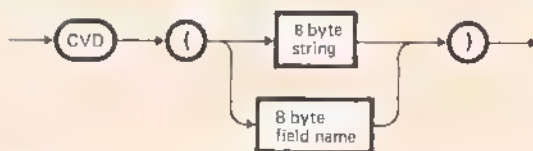



Figura 12-20 Funzione CVD

Esempi

```
1Ø X#=CVD(N$)
2Ø Y!=CVS(R1$)
```

AGGIORNAMENTO DI RECORD DI UN FILE AD ACCESSO DIRETTO

Per aggiornare un file ad accesso diretto, l'utente deve leggere ogni record da aggiornare e quindi registrarlo, come indicato nella tabella che segue.

PASSO	AZIONE
1	L'utente deve aprire il file ad accesso diretto
2	L'utente deve strutturare il buffer in campi

3	L'utente deve leggere il record da aggiornare
4	L'utente deve estrarre i dati dal buffer per visualizzarli o assegnarli a variabili di programma
5	L'utente deve inserire i nuovi valori nei campi del buffer
6	L'utente deve registrare il record aggiornato
7	L'utente può aggiornare un altro record, ripartendo dal passo 3. Diversamente, deve andare al passo 8
8	L'utente deve chiudere il file (a meno che non debba essere riutilizzato con lo stesso metodo d'accesso da un altro programma concatenato)

Esempio

VIDEO

```
LIST
1Ø OPEN "r",1,"1:filetext",128
2Ø FIELD 1,128 AS A$
3Ø INPUT "record number ";RNUM
4Ø GET 1,RNUM
5Ø PRINT A$
6Ø INPUT "give me data ";PP$
7Ø LSET A$="new data --"+PP$
8Ø PUT 1,RNUM
9Ø INPUT "CONTINUE (y/n) ";R$
1ØØ IF R$="y" THEN 3Ø
11Ø CLOSE
Ok
RUN
record number ? 1
new datapoloo
give me data ? gio
CONTINUE (y/n) ? y
record number ? 1
new data --gio
give me data ? pol
CONTINUE (y/n) ? n
Ok
```

COMMENTI

L'istruzione 1Ø apre un file ad accesso diretto, chiamato filetext e residente sul dischetto inserito nell'unità 1

L'istruzione 2Ø, in questo caso, specifica soltanto un nome di campo

L'istruzione 4Ø legge il record da aggiornare, il cui numero è introdotto da tastiera tramite l'istruzione 3Ø

L'istruzione 5Ø visualizza i dati provenienti dal buffer

L'istruzione 7Ø inserisce i nuovi valori nel campo del buffer, concatenando (+) la variabile stringa PP$ alla costante stringa "new data --"

L'istruzione 8Ø registra il record aggiornato

Le istruzioni 9Ø e 1ØØ consentono all'utente di continuare o di fermarsi

L'istruzione 11Ø chiude il file.

13. DEBUGGING E RECUPERO DEGLI ERRORI

SOMMARIO

Questo capitolo descrive le istruzioni ed alcune delle tecniche, che vengono utilizzate per diagnosticare e correggere gli errori di un programma.

INDICE

TIPI DI ERRORE

Succede raramente che un programmatore, per quanto esperto sia, scriva al primo tentativo un programma privo di errori. Se si escludono gli errori relativi all'impostazione di una linea (già descritti nel capitolo 1), gli errori che, in genere, possono essere fatti vengono classificati in due categorie:

- errori che si manifestano durante l'esecuzione di un programma, (run-time errors). Essi interrompono l'esecuzione e determinano la visualizzazione di un messaggio di errore.

- errori logici, che consentono l'esecuzione completa del programma, ma portano a risultati sbagliati o inattesi.

Il processo di individuazione della causa dell'errore (spesso denominato "bug") viene conosciuto con il termine di "debugging". L'M20 fornisce una serie di prestazioni che riducono i costi e le difficoltà di questo processo.

TIPI DI ERRORE	COMMENTI
Run-time errors (errori individuati dall'M20 durante l'esecuzione di un programma o di una linea immediata)	possono essere errori di sintassi (quando una linea contiene una sequenza di caratteri non corretta) o altri tipi di errori di esecuzione (NEXT privo di FOR, RETURN privo di GOSUB, ecc...) E' anche possibile simulare un errore BASIC, o generare un errore definito dall'utente (da gestirsi mediante una routine di error trap). Vedere le istruzioni ERROR e ON ERROR GOTO illustrate di seguito.
Errori di logica (errori che consentono il completamento dell'esecuzione del programma, ma producono risultati errati o inattesi)	sono gli errori più difficili da individuare, e sono denominati errori "logici". Per fornire un semplice esempio, si assuma di aver scritto un programma che deve stampare i risultati di 15 calcoli. Se, a seguito dell'esecuzione del programma, vengono stampati solo 11 risultati, deve essersi verificata una qualche forma di errore. L'individuazione dell'errore, se il programma è lungo e complesso e contiene molti salti, cicli e sottoprogrammi, può risultare un compito non agevole. E' possibile che il controllo non sia stato trasferito all'istruzio-

ne prevista o che qualche calcolo non sia stato effettuato. Le possibilità di errore sono molteplici.

In questi casi, può risultare molto utile poter sapere esattamente quali istruzioni vengono eseguite ed il flusso della loro esecuzione

VISUALIZZAZIONE DEI NUMERI DI LINEA ESEGUITI (TRACING)

Un metodo utile di debugging di errori logici consiste nel tracciare il flusso di esecuzione delle istruzioni per una parte o per l'intero programma. L'M20 fornisce, a questo proposito, due comandi che possono anche essere utilizzati come istruzioni nell'ambito di un programma.

TRON/TROFF (PROGRAMMA/IMMEDIATO)

TRON (TRACE ON) produce una lista dei numeri di linea di tutte le istruzioni eseguite.

TROFF (TRACE OFF) arresta la lista iniziata con l'impostazione del comando TRON.

Figura 13-1 Comando TRON

Figura 13-2 Comando TROFF.

Esempio

VIDEO	COMMENTI
TRON Ok LIST 1Ø K=1Ø 2Ø FOR J=1 TO 2 3Ø L=K+1Ø 4Ø PRINT J;K;L 5Ø K=K+1Ø 6Ø NEXT 7Ø END Ok RUN [1Ø] [2Ø] [3Ø] [4Ø] 1 1Ø 2Ø [5Ø] [6Ø] [3Ø] [4Ø] 2 2Ø 3Ø [5Ø] [6Ø] [7Ø] Ok TROFF Ok	TRON setta l'indicatore di traccia che visualizza il numero di linea di ogni istruzione di programma eseguita. I numeri sono racchiusi tra parentesi quadre. I numeri, che non sono racchiusi tra parentesi quadre (nell'esempio) sono il risultato dell'esecuzione dell'istruzione 4Ø PRINT J;K;L L'indicatore di traccia viene resettato con il comando TROFF oppure all'esecuzione di un comando NEW.

INTERRUZIONE DELL'ESECUZIONE DI UN PROGRAMMA

L'esecuzione di un programma viene interrotta quando:

- si imposta CTRL C, oppure
- viene eseguita un'istruzione STOP, oppure un'istruzione END, oppure
- viene emesso un messaggio d'errore

Nei casi sopra ricordati, l'M20 entra in Stato Comandi (escluso nel caso di un errore di sintassi quando l'M20 entra in Stato Editor).

In Stato Comandi è consentita la visualizzazione delle variabili di programma (con l'uso delle istruzioni immediate PRINT e PRINT USING) o la modifica del loro valore (con l'uso delle istruzioni immediate LET oppure SWAP). E' possibile riprendere l'esecuzione del programma impostando il comando CONT (ad esclusione del caso in cui si è riscontrato un errore o quando l'utente ha modificato il programma).

END (PROGRAMMA)

Fa terminare l'esecuzione del programma, chiude tutti i file dati e riporta l'M20 in Stato Comandi.

Figura 13-3 Istruzione END

Note

Sebbene non sia necessario finire un programma con una istruzione END, questa istruzione è molto utile poichè chiude tutti i file dati aperti; inoltre aumenta la leggibilità del programma. L'istruzione END è anche usata per porre fine al programma al termine di una diramazione (branch). Ad esempio:

```
25Ø IF Z > 1ØØØ THEN END
```

L'istruzione END può essere inserita in qualsiasi parte di un programma per terminarne l'esecuzione. A differenza dell'istruzione STOP, l'istruzione END non produce la visualizzazione del messaggio BREAK.

L'esecuzione di un'istruzione END provoca sempre un ritorno in Stato Comandi. L'utente può visualizzare i valori delle variabili di programma con l'uso dell'istruzione immediata PRINT (o PRINT USING). L'utente può anche riprendere l'esecuzione del programma con il comando CONT però deve tener presente che la END ha chiuso tutti i file dati.

STOP (PROGRAMMA)

Interrompe l'esecuzione del programma e riporta l'M20 in Stato Comandi.

Figura 13-4 Istruzione STOP

Note

L'istruzione STOP, come l'istruzione END, può essere inserita in ogni parte del programma. Quando il controllo dell'esecuzione passa a questa istruzione, il messaggio seguente viene visualizzato:

Break in line nnnnn

A differenza dell'istruzione END, l'istruzione STOP non chiude i file.

Il BASIC ritorna sempre in Stato Comandi dopo l'esecuzione dell'istruzione STOP. L'utente può riprendere l'esecuzione del programma con il comando CONT (vedere in seguito).

Esempio

VIDEO	COMMENTI
LIST 10 INPUT A,B,C 20 X=A*B 30 STOP 40 X=X/C 50 PRINT X 60 END Ok RUN ? 4,3,6 Break in 30 Ok PRINT X 12 Ok CONT 2 Ok	l'istruzione 30 permette di controllare ed osservare il primo valore di X prima che il secondo venga calcolato e visualizzato. In questo semplice caso, l'istruzione STOP non sembra molto utile, ma lo è, invece, nel caso di programma di grosse dimensioni. Impostando un'istruzione STOP alla fine di un trasferimento di controllo, il programma si arresterà solo se il trasferimento viene effettuato. Permette anche di assegnare nuovi valori alle variabili prima che l'esecuzione continui con l'uso del comando CONT. Quando il programma sarà sufficientemente verificato, l'utente potrà cancellare tutte le istruzioni STOP, inserite per il debugging e potrà rinumerare il programma.

CONT (IMMEDIATO)

Permette di riprendere l'esecuzione di un programma interrotto da un CTRL C, o da una istruzione STOP o da una istruzione END.

L'esecuzione riprende dal punto in cui si è verificata l'interruzione.

Figura 13-5 Comando CONT

Caratteristiche

SE...	ALLORA...
CTRL C è stato impostato dopo un prompt di una istruzione INPUT	l'esecuzione continua ristampando il prompt (? seguito da uno spazio, o la stringa prompt)
si incontra una istruzione STOP o una istruzione END	i valori intermedi possono essere esaminati e modificati con l'uso delle istruzioni immediate (PRINT, PRINT USING, LET, SWAP). L'esecuzione può essere ripresa impostando il comando CONT o l'istruzione immediata GOTO, che rimanda il controllo dell'esecuzione ad un numero di linea specificato (l'impostazione di RUN numero di linea invece di GOTO numero di linea consente di ripartire dallo stesso punto, ma provoca l'azzeramento di tutte le variabili del programma). Ad esempio: 10 INPUT A,B,C 20 K=A∧2*5.3:L=B∧3/.26 30 STOP 40 M=C*K+100:PRINT M RUN ? 1,2,3 Break in 30 Ok PRINT L 30.7692

	Ok CONT 115.9 Ok
il programma è stato modificato durante l'interruzione OPPURE si è verificato un errore	CONT non è valido

CONTROLLO E RECUPERO DEGLI ERRORI

Di solito quando si incontra un errore il BASIC arresta l'esecuzione e visualizza un appropriato messaggio di errore. Se questo é un errore di sintassi l'M20 passa in Stato Edit, in tutti gli altri casi l'M20 passa in Stato Comandi.

Spesso l'utente vuole trattare l'errore in modo diverso dallo standard.

Questo può essere fatto scrivendo una routine di trattamento dell'errore.
Usando istruzioni ON ERROR GOTO, è possibile introdurre una o più routine di gestione degli errori. Il controllo viene trasferito alla linea specificata dopo GOTO, quando si é verificato un errore. Una sola routine di trattamento degli errori può essere attiva in un certo istante.L'esecuzione di una istruzione "ON ERROR GOTO Ø"al di fuori di una routine di trattamento dell'errore disattiva il trattamento non standard degli errori.

L'esecuzione di una istruzione "ON ERROR GOTO Ø" all'interno di una routine di gestione degli errori indica che un errore non trattato dalla routine deve essere gestito in modo standard.

Nel caso si verifichi un errore run-time e il trattamento non standard degli errori sia abilitato, il controllo dell'esecuzione viene trasferito alla linea specificata. E' possibile allora fare uso delle funzioni ERR e ERL ed eseguire la routine di recupero dell'errore. La funzione ERR contiene il codice dell'errore, mentre la funzione ERL il numero della linea in cui è stato individuato l'errore.

La routine di gestione degli errori controlla tutti gli errori che l'utente desidera recuperare ed indica il modo di gestirli. Ciò, di solito, porta come conseguenza la correzione dell'errore e la ripresa dell'esecuzione nel punto in cui si era verificato l'errore senza passare in Stato Comandi.

ERROR (PROGRAMMA/IMMEDIATO)

Simula il verificarsi di un errore del linguaggio BASIC, oppure genera un errore definito dall'utente.

Figura 13-6 Istruzione ERROR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numeric expression	il valore dell'espressione numerica rappresenta un codice di errore. Deve essere maggiore di Ø e minore o uguale a 255. Se non è un intero, viene arrotondato all'intero più vicino. Nota: il BASIC non usa tutti i codici d'errore disponibili. I codici non utilizzati visualizzano il messaggio "Unprintable error".

Caratteristiche

SE...	ALLORA...
il valore dell'espressione numerica corrisponde ad un codice di errore del BASIC (vedere l'Appendice F)	l'istruzione ERROR simulerà il verificarsi di tale errore, ed il messaggio di errore corrispondente verrà visualizzato. Ad esempio: LIST 1Ø S=1Ø 2Ø T=5 3Ø ERROR S+T 4Ø END Ok RUN String too long in line 3Ø Ok Oppure, in stato immediato ERROR 15 String too long Ok
il valore dell'espressione numerica è maggiore di tutti i codici di errore del BASIC	l'istruzione ERROR genera un errore definito dall'utente. Questo codice di errore può allora essere convenientemente usato in una routine di gestione degli errori (vedere ON ERROR GOTO qui di seguito). Nota: Per definire i propri codici di errore si usi un valore che è maggiore di tutti i codici di errore BASIC (E' preferibile usare codici prossimi a 255 -valore massimo-, così da mantenere la compatibilità nell'eventualità in cui più codici di errore siano aggiunti al linguaggio BASIC)
un'istruzione ERROR specifica un codice per il quale non è stato definito un messaggio di errore	il BASIC risponde con il messaggio: Unprintable error

ON ERROR GOTO (PROGRAMMA)

Permette la gestione degli errori e specifica la prima linea della routine di gestione degli errori (ogni programma BASIC può avere solo una routine di gestione degli errori attiva in un dato istante).

Figura 13-7 Istruzione ON ERROR GOTO

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
line number	è la prima linea di una routine di gestione degli errori. Deve essere maggiore di Ø e minore o uguale a 65529. Nota: L'istruzione ON ERROR GOTO Ø non trasferisce il controllo alla linea di numero Ø, in caso di errore, ma disabilita la gestione non standard degli errori. Così se l'istruzione ON ERROR GOTO Ø è in una routine di gestione degli errori e questa istruzione viene eseguita quando un errore è ancora pendente allora viene visualizzato il messaggio standard di errore ed il sistema entra in Stato Comandi.

Esempio

VIDEO	COMMENTI
. . . 110 ON ERROR GOTO 400 120 INPUT "WHAT IS YOUR BET";B 130 IF B>5000 THEN ERROR 210 . . . 400 IF ERR=210 THEN PRINT "HOUSE LIMIT IS $5000" 410 IF ERL=130 THEN RESUME 120 420 ON ERROR GOTO 0 . . .	Se si introduce un valore di B maggiore di 5000, viene visualizzato il messaggio: HOUSE LIMIT IS $5000 e l'esecuzione riprende alla linea 120. Se si incontrano altri errori, l'istruzione 420 fa si che sia visualizzato il messaggio standard di errore.

Caratteristiche

SE...	ALLORA...
E' stata abilitata la gestione non standard degli errori (tramite una istruzione ON ERROR GOTO)	tutti gli errori individuati (inclusi gli errori verificatisi nelle linee ad esecuzione immediata) determineranno un trasferimento alla routine di gestione dell'errore specificata
il numero di linea dopo ON ERROR GOTO non esiste	viene visualizzato il messaggio: Undefined line
viene eseguita l'istruzione ON ERROR GOTO 0	la gestione non standard degli errori viene disabilitata. Errori successivi visualizzeranno un messaggio standard di errore ed interromperanno l'esecuzione.
l'istruzione ON ERROR GOTO 0 é inserita in una routine di gestione degli errori	il programma BASIC si arresta e viene visualizzato il messaggio standard di errore per l'errore che ha causato l'attivazione della routine.

	Nota: Si raccomanda che tutte le routine di gestione degli errori eseguano una istruzione ON ERROR GOTO Ø nel caso in cui si verifichi un errore di cui non è previsto il recupero.
si verifica un errore durante l'esecuzione di una routine di gestione non standard degli errori	il programma BASIC evidenzia un messaggio di errore e l'esecuzione termina. Non è possibile trasferire il controllo alla routine di gestione degli errori quando un errore si verifica nell'ambito della stessa routine.

Nota

Non è consentito gestire in modo non standard gli errori di "overflow" e "division by zero".

ERL/ERR (PROGRAMMA/IMMEDIATO)

Quando si verifica un errore, la funzione ERL fa ritornare il numero di linea nella quale l'errore è stato individuato, mentre la funzione ERR fa ritornare il codice di errore.

Figura 13-8 Funzione ERL

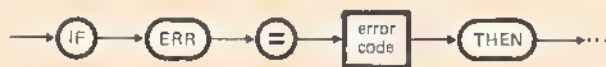



Figura 13-9 Funzione ERR

Caratteristiche

Le funzioni ERL e ERR vengono generalmente, utilizzate nelle istruzioni IF...THEN...ELSE oppure IF...GOTO...ELSE per trasferire il controllo d'esecuzione alla routine di gestione degli errori.

SE...	ALLORA...
l'istruzione che ha causato l'errore è stata una istruzione immediata	ERL conterrà il valore 65535. Per verificare se si è verificato un errore in una istruzione immediata si usi: IF 65535=ERL THEN... Altrimenti, si usi: IF ERR = error code THEN... IF ERL = line number THEN...
line number non è sul lato destro dell'operatore di confronto	non può essere attribuito un nuovo numero di linea mediante la funzione RENUM

Esempio

VIDEO	COMMENTI
LIST 1Ø REM RETTANGOLO2 2Ø ON ERROR GOTO 7Ø 3Ø INPUT "Base e Altezza";B,H 4Ø IF (B<Ø) OR (H<Ø) THEN ERROR 2ØØ 5Ø PRINT "Area="; B*H;" B=";B;" H=";H 6Ø GOTO 3Ø 7Ø IF (ERR=2ØØ) AND (ERL =4Ø) THEN PRINT "B o H<Ø":RESUME 3Ø 8Ø ON ERROR GOTO Ø 9Ø END Ok RUN Base e Altezza? -2,5	Se si imposta un valore negativo per B oppure per H, la routine di gestione dell'errore viene attivata ed il sistema visualizza: B o H<Ø L'esecuzione viene ripresa all'istruzione 3Ø (vedere l'istruzio-

```
B o H < Ø
Base e Altezza? 2,5
Area= 1Ø B= 2 H= 5
Base e Altezza?∧C
Break in 3Ø
Ok
```

ne RESUME qui di seguito).

Si noti l'uso di ERR e di ERL nella routine di gestione degli errori

Note

Queste funzioni possono essere usate anche come normali funzioni BASIC.

Per esempio:

```
PRINT ERR
PRINT "Troppo grande", ERL
I% = ERR
```

RESUME (PROGRAMMA)

Riprende l'esecuzione del programma principale dopo aver eseguito una routine di gestione degli errori.

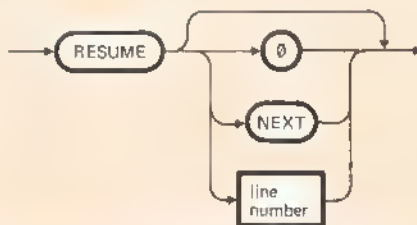



Figura 13-10 Istruzione RESUME

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
Ø	l'esecuzione riprende all'istruzione che ha causato l'errore. Nota: RESUME Ø e RESUME sono equivalenti
NEXT	l'esecuzione riprende alla prima istruzione successiva a quella che ha causato l'errore
line number	l'esecuzione riprende al numero di linea specificato

Note

Un'istruzione RESUME che non è inclusa nella routine di gestione degli errori fa visualizzare il messaggio "RESUME without error".

Esempi

VIDEO	COMMENTI
LIST 1Ø REM RETTANGOLO3 2Ø ON ERROR GOTO 7Ø 3Ø INPUT "Base e Altezza";B,H 4Ø IF (B<Ø) OR (H<Ø) THEN ERROR 2ØØ 5Ø PRINT "Area=";B*H;"B=";B;" H=";H 6Ø GOTO 3Ø 7Ø IF (ERR=2ØØ) AND (ERL=4Ø) THEN RESUME 8Ø ON ERROR GOTO Ø 9Ø END Ok RUN Base e Altezza? -2,5 ^C Ok	se si introduce un valore negativo per B oppure H, la routine di gestione degli errori viene attivata. In questo caso la routine determina la ripresa dell'esecuzione all'istruzione che ha causato l'errore, causando in questo modo un ciclo senza fine. Per fermare l'esecuzione agire su CTRL C

```
7Ø IF (ERR=2ØØ) AND (ERL=4Ø) THEN RESUME NEXT
RUN
Base e Altezza? -2,5
Area=-1Ø B=-2 H= 5
Base e Altezza? ^C
Break in 3Ø
Ok
```

correggendo così la linea 7Ø, l'errore viene ignorato

```
7Ø IF (ERR=2ØØ) AND (ERL=4Ø) THEN RESUME 3Ø
RUN
Base e Altezza? -2,5
Base e Altezza? 2,5
Area= 1Ø B= 2 H= 5
Base e Altezza? ^C
Break in 3Ø
Ok
```

correggendo così la linea 7Ø la routine di gestione degli errori fa sì che l'esecuzione sia ripresa alla linea 3Ø

14. GRAFICA

SOMMARIO

Questo capitolo intende far conoscere all'utente le prestazioni grafiche disponibili sull'M20 e le loro modalità d'uso.

In un "computer" la grafica è il mezzo con il quale l'informazione viene rappresentata sotto forma di figura; una qualunque combinazione di testi e di figure è "grafica".

INDICE

INTRODUZIONE

L'M20 è disponibile in due versioni: con video in bianco e nero e con video a colori. In entrambi i casi l'utente può visualizzare una matrice di 512 x 256 "pixel" oppure 480 x 256 "pixel", dove il termine "pixel è l'abbreviazione di "picture element" (elemento di figura). Un elemento di figura è un punto luminoso del video e una riga di punti luminosi costituisce una "scanline".

La scelta dell'una o dell'altra modalità di visualizzazione viene fatta dall'utente tramite il comando SSYS del PCOS oppure tramite l'istruzione WINDOW (per variare lo spazio tra linee e caratteri).

Le dimensioni massime dell'immagine su video sono: 225 mm in orizzontale e 140 mm in verticale.

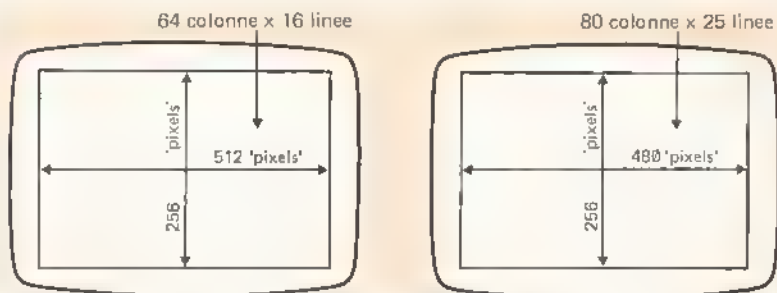



Figura 14-1 Modalità di visualizzazione (512 x 256 oppure 480 x 256)

Le linee di testo che compaiono sul video possono contenere un numero massimo di 64 o 80 caratteri. Anche lo spazio tra una linea di testo e la successiva può variare; sul video può esserci da un minimo di 16 linee fino ad un massimo di 25.

Un M20 con video a colori può essere un sistema a 4 oppure a 8 colori. Nella versione a 4 colori, i caratteri e la grafica possono essere visualizzati usando quattro colori, scelti tra 8 possibili (tramite l'istruzione COLOR - Per selezionare i colori contemporanei).

Nella versione a 8 colori i caratteri e la grafica possono essere visualizzati usando tutti gli otto colori contemporaneamente.

Gli otto colori possibili sono: nero, verde, blu, cyan, rosso, giallo, magenta, e bianco.

In una versione a colori, il colore di fondo del video (detto anche colore di background) è nero e il colore di primo piano con cui l'utente visualizza i caratteri e la grafica (detto anche colore di foreground) è di norma verde.

Nella versione in bianco e nero, invece, il colore di background è nero ed il colore di foreground è bianco.

Con qualsiasi tipo di video è possibile variare questi due colori, con l'uso dell'istruzione COLOR - Per selezione i colori di foreground e di background. Ovviamente per il video in bianco e nero è solo possibile scambiare i due colori.

In entrambe le versioni dell'M20, il video può essere suddiviso dall'utente in aree rettangolari, chiamate finestre ("windows").

FINESTRE

Su una finestra, l'utente può operare esattamente come sull'intero video; su essa può eseguire sia operazioni di grafica sia di testo e possono quindi coesistere sia figure sia linee di testo. Le operazioni eseguite su finestre diverse sono completamente indipendenti.

Le dimensioni delle finestre vengono stabilite con l'istruzione WINDOW (per aprire una finestra) descritta in seguito.

Se viene fatto un tentativo di disegnare una figura o una stringa di caratteri di date dimensioni e orientamento (tramite il comando LABEL del PCOS) e se questa figura o questa stringa cadono parzialmente fuori dalla finestra sulla quale si opera (finestra attuale), solo la porzione che cade dentro la finestra viene tracciata e la parte esterna viene ignorata ("clipped").

Supponiamo di eseguire operazioni di grafica; in questo caso l'utente può operare definendo un proprio sistema di coordinate ("coordinate utente") adatto al problema da trattare (vedere l'istruzione SCALE) oppure usare il sistema di coordinate di default (vedere il paragrafo "Sistemi di Coordinate").

Nel sistema di coordinate di default l'origine è posta nell'angolo inferiore sinistro della finestra, il semi-asse positivo delle x va dall'origine verso destra; il semi-asse positivo delle y va dall'origine verso l'alto e la finestra viene suddivisa in 512 unità lungo l'asse delle x e 256 unità lungo l'asse delle y. Le coordinate di ogni punto

sono quindi espresse in coordinate utente, a meno che si operi sull'intero video - non suddiviso in finestre - e sia stata scelta la modalità di visualizzazione 512 x 256 (cioè 64 colonne x 16 linee). In questo caso soltanto le coordinate di ogni punto sono coordinate "hardware", cioè espresse in pixel. Operando con coordinate hardware è possibile individuare un pixel specificando le sue coordinate; operando con coordinate utente è possibile soltanto individuare il pixel che è il più vicino alle coordinate specificate.

Supponiamo di eseguire operazioni su linee di testo. In questo caso l'origine (ovvero la posizione in cui verrà visualizzato il primo carattere impostato) coincide con l'angolo superiore sinistro della finestra. Le coordinate di ogni carattere di testo vengono, sempre, espresse in termine di numero di riga e di numero di colonna, quindi l'angolo superiore sinistro corrisponde alle coordinate (1,1). E' possibile, però, iniziare a visualizzare un carattere da qualunque altra posizione sul video con l'uso dell'istruzione CURSOR (descritta in seguito).

Le coordinate grafiche sono completamente indipendenti dalle coordinate di testo.

Ogni finestra, in cui il video è stato suddiviso ha due cursori: il cursore di testo e il cursore grafico. Entrambi i cursori possono essere posizionati e visualizzati dall'utente in qualunque posizione del video con l'uso dell'istruzione CURSOR.

Si osservi, però, che il cursore di testo si sposta automaticamente di una posizione ogni volta che un carattere viene impostato; mentre, quando viene eseguita un'istruzione di grafica il cursore grafico non si sposta.

Quando l'utente entra in ambiente BASIC, è presente una sola finestra (l'intero video, cioè la finestra numero 1) e il sistema di coordinate è quello di default. L'utente può definire (aprire) una nuova finestra come una porzione rettangolare di una qualsiasi finestra già esistente.

Per fare ciò, l'utente deve usare l'istruzione WINDOW - Per aprire una finestra. Il numero massimo di finestre aperte contemporaneamente è 16. Il comando SBASIC del PCOS può allocare spazio in memoria per un prefissato numero di finestre.

WINDOW - Per aprire una finestra (PROGRAMMA/IMMEDIATO)

Apre una nuova finestra, suddividendo la finestra attuale ("current window"). Per finestra attuale si deve intendere la finestra sulla quale l'utente si trova ad operare.

Quando si apre una nuova finestra, la finestra attuale che l'ha generata è anche detta "parent window" (finestra genitrice). L'istruzione apre, ma non seleziona, una nuova finestra (vedere WINDOW - Per selezionare una finestra).

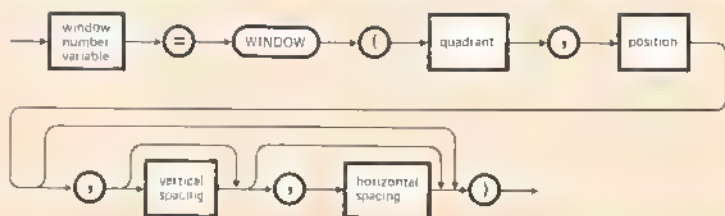



Figura 14-2 Istruzione WINDOW - Per aprire una finestra

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number variable	è una variabile intera, alla quale il sistema operativo assegna un valore, che identifica la finestra da aprire. Questo valore è un intero da 2 a 16. I valori vengono assegnati dal sistema in ordine crescente. La finestra che inizialmente coincide con l'intero video viene identificata con il numero 1, la prima che viene aperta dall'utente con il numero 2 ecc.

Supponiamo, ad esempio, che l'utente stia operando sulla finestra identificata dal numero 4 e decida di aprirne un'altra: a questa nuova finestra il sistema assegnerà il numero 5, a meno che una finestra con il numero 2 oppure 3 sia stata chiusa (v. istruzione CLOSE WINDOW) perché, in tal caso, alla nuova finestra verrà assegnato il più piccolo numero disponibile. La finestra 1 non può mai essere chiusa.

La finestra che ha dato luogo alla nuova finestra viene chiamata la finestra "genitrice" ("parent window"). In questo esempio la finestra 4 viene detta finestra genitrice.

Nota: Il sistema considera l'intero video come la finestra principale e, pertanto, gli assegna il numero 1, che la identifica sempre, anche se in seguito viene suddivisa in altre finestre.

quadrant

specifica dove, all'interno della finestra "genitrice", la nuova finestra deve essere aperta.

L'utente dispone di quattro possibilità:

Ø nella parte alta della finestra "genitrice"
1 nella parte bassa
2 nella parte sinistra
3 nella parte destra

position

individua la posizione, in corrispondenza della quale la finestra "genitrice" viene suddivisa per dar luogo ad una nuova finestra

Se il valore del parametro 'quadrant' è Ø oppure 1, la suddivisione è orizzontale. In questo caso, il parametro 'position' è un numero intero di "scanline" compreso nell'intervallo, i cui estremi sono:

limite inferiore = 1

limite superiore = 255

Nota: La posizione della linea di suddivisione (che non viene visualizzata) è, sempre, calcolata a partire dall'alto della finestra genitrice.

Se "quadrant" vale 2 oppure 3, la suddivisione è verticale. In questo caso, il parametro "position" è un numero intero di caratteri, compreso nell'intervallo, i cui estremi sono:

limite inferiore = 1

limite superiore = larghezza della finestra genitrice, diminuita di una unità.

Se "quadrant" vale 2 oppure 3, la linea di suddivisione (che non viene visualizzata) è calcolata a partire dal bordo sinistro della finestra genitrice.

Nota: se "position" è uguale a 1, la finestra genitrice viene suddivisa a metà (verticalmente oppure orizzontalmente, in funzione del valore di quadrant).

vertical spacing

è un parametro opzionale, che stabilisce il numero di "scanline" per ogni linea di testo della finestra da aprire. Ha, come valore minimo, 10 (sull'intero video vengono visualizzate 25 linee di testo) e come valore massimo 16 (16 linee di testo).

Se vengono omessi sia il parametro vertical spacing sia il parametro horizontal spacing, viene assunto il vertical spacing della finestra genitrice. Se invece viene omesso solo il parametro vertical spacing ed il parametro horizontal spacing è diverso da quello della finestra genitrice, allora il vertical spacing viene assunto pari a 16 se l'horizontal spacing è 8 e pari a 10 se l'horizontal spacing è 6.

horizontal spacing	parametro opzionale che stabilisce lo spazio che deve essere presente fra due caratteri centigui in una linea di testo (per la finestra da aprire). Viene espresso in termine di numero di 'pixel' e può assumere due valori : 6 oppure 8. ll primo di questi due valori consente un massimo di 8Ø caratteri per una linea completa di testo (sull'intero video), il secondo 64. Per default, horizontal spacing assume il valore del parametro omonimo della finestra genitrice.

Note

Quando una finestra viene aperta il precedente contenuto di quell'area di videe viene cancellate e, come colori di background e di foreground, vengono assunti quelli della finestra genitrice.

Esempi

SE l'utente imposta...	ALLORA...
A=WINDOW (Ø,1ØØ) **CR**	la suddivisione è orizzontale, la nuova finestra viene ad essere la parte alta della finestra genitrice. L'altezza della finestra da aprire è 1ØØ scanline. l parametri horizontal spacing e vertical spacing assumone i valori di default (quelli della finestra genitrice).

```
A=WINDOW(Ø,1ØØ,14) CR
```

come sopra, con l'unica eccezione che lo spazio fra due linee contigue di testo è di 14 scanline.

```
B=WINDOW(2,5Ø) CR
```

la suddivisione è verticale, la nuova finestra viene ad essere la parte sinistra della finestra genitrice. La larghezza della finestra da aprire è 5Ø posizioni di carattere.

I parametri horizontal spacing e vertical spacing assumono i valori di default.

```
A=WINDOW(Ø,1ØØ) CR
PRINT A CR
```

l'istruzione WINDOW apre una finestra, che l'utente identifica con la variabile A. Il sistema operativo assegna un valore intero a questa variabile.Il contenuto di A viene, poi, visualizzato con l'uso dell'istruzione PRINT A.

```
A=3:D=4Ø:F=15:G=8 CR
W=WINDOW(A,D,F,G) CR
```

la suddivisione è verticale. La nuova finestra è la parte destra della finestra "genitrice". La lunghezza della finestra da aprire è 5Ø posizioni di carattere.

Il parametro vertical spacing vale 15 scanline ed ogni linea completa di testo può contenere 64 caratteri.

```
W2=WINDOW(1,184,16) CR
W3=WINDOW(Ø,5Ø,16)  CR
W4=WINDOW(2,16,16)  CR
W5=WINDOW(3,43,16)  CR
```

queste istruzioni WINDOW suddividono il video in 5 finestre. Nella figura qui di seguito riportata viene indicata la sequenza di questa suddivisione

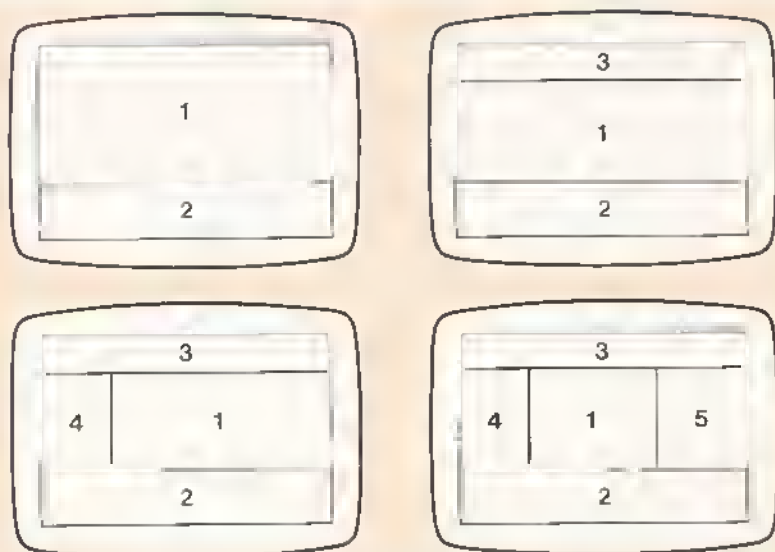



Figura 14-3 Sequenza con la quale vengono aperte le finestre

Note

Si è visto come l'utente possa specificare, al momento dell'apertura di una finestra, sia lo spazio verticale fra linee contigue di testo, sia lo spazio orizzontale fra caratteri contigui su una stessa linea.

In alcuni casi, può risultare necessario variare questi due valori relativamente ad una finestra, che è già stata aperta. Questo può essere fatto con l'uso della istruzione WINDOW, ponendo a zero i parametri quadrant e position.

WINDOW - Per variare lo spazio tra linee e caratteri (PROGRAMMA/IMMEDIATO)

Con questa istruzione non viene aperta alcuna nuova finestra. Viene usata, semplicemente, per variare lo spazio fra caratteri e/o lo spazio fra linee di testo della finestra attuale.

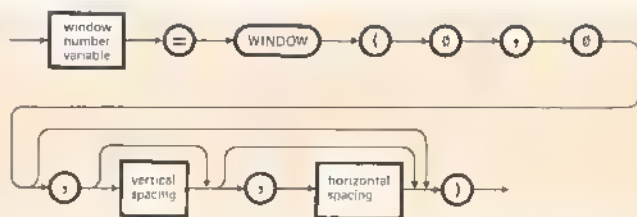



Figura 14-4 Istruzione WINDOW - Per variare lo spazio tra linee e caratteri

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number variable	è una variabile intera, alla quale il sistema assegna il numero della finestra attuale.
Ø	valore del parametro quadrant
Ø	valore del parametro position
vertical spacing	è un parametro opzionale, che stabilisce il numero di scanline per ogni linea di testo della finestra attuale. Ha come valore minimo 1Ø (sull'intero video vengono visualizzate 25 linee di testo) e come valore massimo 16 (16 linee di testo). Se vengono omessi sia il vertical spacing sia l'horizontal spacing, lo spazio tra le linee rimane immutato. Se viene omesso solo il vertical spacing e l'horizontal spacing ha un valore diverso dal valore in vigore per la finestra attuale, il vertical spacing sarà 16 se l'horizontal spacing è 8 e sarà 1Ø se l'horizontal spacing è 6.

horizontal spacing	è un parametro opzionale, che stabilisce lo spazio fra due caratteri contigui in una linea di testo. Viene espresso in numero di 'pixel' e può assumere due valori: 6 oppure 8. Il primo di questi due valori consente un massimo di 80 caratteri per una linea completa di testo; il secondo 64. Se omesso, lo spazio fra caratteri rimane immutato.

Note

Al momento dell'accensione dell'M20, è presente una sola finestra, che coincide con l'intero video e che ha numero 1. Il parametro horizontal spacing ha valore 8 ed il parametro vertical spacing ha valore 16, così sull'intero video può essere visualizzato un numero massimo di 16 linee di testo, con 64 caratteri per linea (nell'ipotesi di usare un PCOS standard).

L'utente può variare questa modalità di visualizzazione ed avere 25 linee di testo con 80 caratteri ciascuna sia usando il comando SSYS del PCOS, sia usando l'istruzione WINDOW. In quest'ultimo caso, dopo essere entrato in BASIC, l'utente deve usare l'istruzione WINDOW (per variare lo spazio tra linee e caratteri) e porre il parametro horizontal spacing uguale a 6 ed il parametro vertical spacing uguale a 10.

E' possibile solo avere 64 oppure 80 colonne, ma è possibile variare il numero di righe da un minimo di 16 a un massimo di 25, sempre tramite questa istruzione WINDOW, variando opportunamente il parametro vertical spacing da 16 a 10.

WINDOW - Per selezionare una finestra (PROGRAMMA/IMMEDIATO)

Seleziona la finestra su cui operare. La finestra selezionata diviene la finestra attuale.

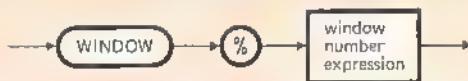



Figura 14-5 Istruzione WINDOW - Per selezionare una finestra

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è una espressione numerica, il cui valore, arrotondato all'intero più vicino, identifica una finestra già aperta in precedenza e la seleziona. Il parametro window number expression può assumere solo valori interi da 1 a 16 e deve corrispondere ad una finestra esistente, altrimenti si ha una segnalazione d'errore.

Esempi

SE l'utente imposta...	ALLORA...
WINDOW %A CR	il sistema seleziona la finestra identificata dal valore intero, contenuto nella variabile A. Se questo valore è noto (ad es.2), l'istruzione WINDOW può essere impostata anche nel modo seguente: WINDOW %2 CR
WINDOW %1 CR	il sistema seleziona la finestra identificata dal numero 1. Come noto, questa è la finestra principale (cioè l'intero video o la finestra risultante per successive suddivisioni)

CLOSE WINDOW (PROGRAMMA/IMMEDIATO)

Chiude una finestra specificata o tutte le finestre aperte.

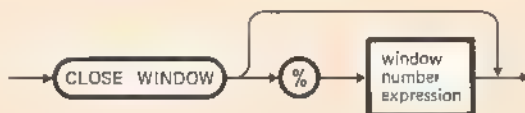



Figura 14-6 Istruzione CLOSE WINDOW

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è una espressione numerica, arrotondata all'intero più vicino, il cui valore identifica la finestra da chiudere. Se window number expression viene omesso il sistema ritorna allo "stato iniziale" (una sola finestra: l'intero video).

Caratteristiche

L'istruzione CLOSE WINDOW, se eseguita con il parametro window number expression, chiude la finestra identificata dal parametro.

Il sistema assegna l'area della finestra che viene chiusa al rettangolo che era stato suddiviso al momento della sua apertura e quest'area viene ad avere il colore di background della finestra a cui viene assegnata.

Nota

L'istruzione CLOSE WINDOW, se fatta operare sulla finestra principale, non produce alcun effetto. La finestra numero 1 (o finestra principale) non può mai essere chiusa.

USO DEL COLORE

Come predetto in M20 con video a colori può essere un sistema a 4 oppure a 8 colori, a seconda che i colori contemporanei possono essere 4 oppure 8.

Il video a colori è comunque lo stesso: si passa da 4 a 8 colori aggiungendo una piastra di memoria e variando opportunamente alcuni ponticelli.

Con un video in bianco e nero esiste in memoria una sola matrice di bit (Bit-Map) corrispondente ai pixel del video, per cui ogni pixel corrisponde a un bit della Bit-Map.

Con un video a 4 colori esistono in memoria due matrici di bit sovrapposte, per cui ogni pixel corrisponde a due bit della Bit-Map. Quindi 4 colori possono essere associati a ogni pixel, dato che due bit possono generare i numeri da Ø a 3.

Nella versione a 4 colori l'utente ha la possibilità di visualizzare i pixel con 4 colori scelti tra gli 8 possibili, tramite l'istruzione COLOR (per selezionare i colori contemporanei).

Con un video a 8 colori esistono in memoria tre matrici di bit sovrapposte, per cui ogni pixel corrisponde a tre bit della Bit-Map. Quindi 8 colori possono essere associati a ogni pixel, dato che tre bit possono generare i numeri da Ø a 7.

Nella versione a 8 colori l'utente ha la possibilità di visualizzare i pixel con tutti gli 8 colori a disposizione.

Codici di Colore

La tabella, riportata qui di seguito, elenca i colori dell'M20 e i corrispondenti codici di colore ("colour codes"). Questa tabella è valida sia per un sistema a quattro che per un sistema a otto colori.

CODICE DI COLORE	COLORE
Ø	nero
1	verde
2	blu
3	cyan (turchese)
4	rosso
5	giallo
6	magenta
7	bianco

Tabella 14-1

Numeri di Colore

Molte istruzioni di grafica hanno un parametro detto "numero di colore".

I numeri di colore coincidono con i codici di colore in un sistema a 8 colori.

In un sistema a 4 colori, invece, i numeri di colore non sono sinonimi dei codici di colore, ma ogni numero viene associato a un codice di colore tramite l'istruzione COLOR (per selezionare i colori contemporanei). L'associazione (o "mapping") a viene in questo modo: il numero di colore può solo essere un numero intero tra Ø a 3 e rappresenta la posizione che è stata assegnata a un codice di colore nell'istruzione COLOR predetta: essa specifica sempre, alla destra del segno di uguale, 4 codici di colore scelti tra gli 8 possibili (il primo codice da sinistra viene associato al numero di colore Ø, il secondo al numero 1 ecc.). L'istruzione COLOR predetta non ha alcun effetto né per un sistema a 8 colori (dato che i numeri di colore coincidono con i codici di colore) nè per un sistema con video bianco e nero, dove il numero di colore è un intero che può solo essere Ø (a rappresentare il nero) oppure 1 (a rappresentare il bianco).

Colori di Default

In un sistema a 4 colori, se l'istruzione COLOR (per selezionare i colori contemporanei) non viene eseguita, il sistema assume 4 colori di default che sono il nero, il verde, il blu, e il rosso. Si noti che è come se fosse stata eseguita l'istruzione:

```
COLOR = Ø, 1, 2, 4
```

In un sistema a 8 colori non esistono colori di default, dato che tutti gli 8 colori sono sempre disponibili.

Colori di Background e di Foreground

Ogni finestra ha un colore di background (cioè un colore di fondo) e un colore di foreground (cioè di primo piano, ovvero il colore con cui si visualizzano i caratteri e le figure, tracciate con istruzioni di grafica che non specificano un diverso numero di colore).

I colori di background e di foreground possono essere scelti dall'utente o assunti per default.

I valori di default (sia per un video bianco e nero, sia per un video a colori) corrispondono ai numeri di colore Ø (background) e 1 (foreground). A questi numeri sono associati i colori:

- nero (background) e bianco (foreground) per un video bianco e nero;

- nero (background) e verde (foreground) per un video a 8 colori;

- nero (background) e verde (foreground) per un video a 4 colori, se non viene eseguita una istruzione COLOR (per la selezione dei colori contemporanei).

  Se invece viene eseguita detta istruzione, i colori di background e foreground sono rispettivamente il primo e il secondo colore specificati nell'istruzione.

L'utente può specificare numeri di colore di background e di foreground diversi dai numeri di colore di default tramite l'istruzione COLOR (per selezionare i colori di foreground e di background). Con un video bianco e nero l'utente può solo invertire i due valori di default e avere quindi il bianco come colore di fondo e il nero come colore di primo piano.

Colore del Cursore

Ogni finestra ha due cursori: uno di testo e uno grafico (vedere paragrafo "Visualizzazione dei Cursori").

Nella versione a 4 colori il colore del cursore (sia di testo che grafico) corrisponde all'ultimo codice di colore nell'istruzione COLOR (per selezionare i colori contemporanei) oppure è il rosso, se detta istruzione non viene eseguita. Infatti, in assenza di detta istruzione, è come se fosse stata eseguita l'istruzione:

```
COLOR = Ø, 1. 2, 4
```

Nelle versione a 8 colori il colore del cursore (sia di testo che grafico) è sempre il bianco.

E' possibile cambiare la forma del cursore, sia grafico sia di testo, con l'istruzione CURSOR, me non è possibile cembierne il colore (se non per un sisteme e 4 colori). L'unico modo per cembiare il colore del cursore è, limitatemente a un sisteme a 4 colori, quello di eseguire un'eltra istruzione COLOR con un diverso codice di colore in querta posizione.

COLOR - Per selezionare i colori contemporanei (PROGRAMMA/IMMEDIATO)

Seleziona i quattro colori contemporanei tra gli otto possibili (in un sistema con video a 4 colori).

Con un video bianco e nero o con un video a otto colori questa istruzione, se usata, non ha alcun effetto.

Figura 14-7 Istruzione COLOR - Per selezionare i 4 colori contemporenei

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
colour code	è una espressione numerica, che assume un valore intero da Ø e 7. A ciascuno di questi valori corrisponde un colore, come indicato nelle Tabella 14-1. Se l'espressione numerica non dè risultato intero, questo viene errotondato all'intero più vicino.

COLOR - Per selezionare i colori di foregound e di background (PROGRAMMA/IMMEDIATO)

Seleziona i colori di foreground e di background per una finestra specificata.

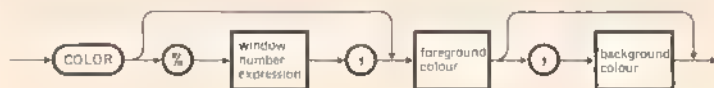



Figura 14-8 Istruzione COLOR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è una espressione numerica il cui valore, arrotondato all'intero più vicino, seleziona la finestra su cui operare. E' opzionale. Se omessa, il sistema opera sulla finestra attuale.
foreground colour	è un numero di colore, che individua e seleziona il colore di foreground
background colour	è un numero di colore, che individua e seleziona il colore di background. Se omesso, il precedente colore di background rimane immutato.

Caratteristiche

Sia nella versione con video bianco e nero, sia nelle versioni con video a 4 o a 8 colori, i numeri di colore sono Ø per il colore di background (di default) e 1 per il colore di foreground (di default).

I colori di background e di foreground di una finestra possono essere variati con l'uso della predetta istruzione COLOR per tutti i tipi di video.

Questa variazione non è di immediata visualizzazione e l'utente, per rendersene conto, dopo l'esecuzione dell'istruzione COLOR, deve effettuare una delle seguenti operazioni:

1. Eseguire l'istruzione CLS (descritta in seguito, nel capitolo): in questo modo la finestra selezionata assume il nuovo colore di background.

2. Eseguire l'istruzione PRESET (descritta, in seguito, nel capitolo): con l'uso ripetuto di questa istruzione, pixel della finestra selezionata assumono il nuovo colore di background.

3. Visualizzare un testo: la parte del video, nella quale il testo viene visualizzato, assume i nuovi colori di background e di foreground.

Esempi

SE l'utente imposta...	ALLORA...
COLOR Ø,1 **CR**	la finestra attuale ha il bianco come colore di background e il nero come colore di foreground
COLOR %A,Ø,1 **CR**	come sopra, ma l'istruzione COLOR opera sulla finestra identificata dalla variabile A.

Note

Se l'utente imposta COLOR Ø.1 invece di COLOR Ø,1, i successivi caratteri impostati non risultano visibili (dato che Ø.1 viene arrotondato a Ø). Per far nuovamente visualizzare i caratteri, l'utente deve impostare CLEAR oppure l'istruzione COLOR predetta in modo corretto.

CLS (PROGRAMMA/IMMEDIATO)

Cancella il contenuto di una finestra selezionata o della finestra attuale e la colora con il colore di background.

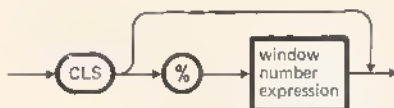



Figura 14-9 Istruzione CLS

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è una espressione numerica arrotondata all'intero più vicino, il cui valore identifica la finestra su cui operare. L'uso di questo parametro è opzionale. Se non viene specificato, l'operazione viene eseguita sulla finestra attuale.

SISTEMI DI COORDINATE

L'istruzione SCALE consente di scegliere il sistema di coordinate utente più idoneo al problema in esame, definendo la trasformazione tra il sistema di coordinate di default e il sistema di coordinate prescelto.

Il sistema di coordinate di default è il sistema di coordinate "hardware" (in pixel) solo se il video non è stato suddiviso in finestre ed è stata scelta la modalità di visualizzazione 512 x 256 (cioè 64 colonne x 16 linee), oppure se si è in ambiente PCOS.

In tutti gli atri casi il sistema di coordinate di default è pur sempre un sistema di coordinate utente, nel senso che la finestra viene automaticamente suddivisa in 512 unità lungo l'asse delle x e in 256 unità lungo l'asse delle y e çhe l'origine degli assı è nell'angolo inferiore sinistro della finestra.

Si ricordi che, operando con un sistema di coordinate hardware, è possibile individuare un pixel specificando le sue coordinate; operando invece con un sistema di coordinate utente è possibile soltanto individuare il pixel che è il più vicino alle coordinate specificate.

Le funzioni SCALEX e SCALEY permettono di ottenere le coordinate hardware di un punto qualsiasi del video.

l valore di ritorno di SCALEX esprime la misura in pixel del segmento lungo l'asse x, compreso tra l'origine della finestra attuale (angolo in basso a sinistra) e il punto specificato.

Il valore di ritorno di SCALEY esprime la misura in pixel del segmento lungo l'asse y, compreso tra l'origine della finestra attuale (angolo in basso a sinistra) e il punto specificato.

L'utilizzo delle funzioni SCALEX e SCALEY può essere utile in vari casi. Per esempio è possibile, tramite l'istruzione SCALE e le funzioni SCALEX e SCALEY, individuare in modo esatto un singolo pixel anche se si opera con un sistema di coordinate utente (cioè se si opera su una finestra che non coincida con l'intero video o sull'intero video con modalità di visualizzazione 480 x 256). Per far ciò è sufficiente operare la trasformazione di coordinate:

```
SCALE  Ø,SCALEX(511),Ø,SCALEY(255)
```

Dopo questa trasformazione infatti qualsiasi valore intero compreso tra Ø e SCALEX(511) individua l'ascissa di un pixel e qualsiasi valore intero tra Ø e SCALEY(255) individua l'ordinata di un pixel della finestra attuale.

Un altro utilizzo tipico delle funzioni SCALEX e SCALEY si ha con il comando LABEL del PCOS (richiamabile da BASIC tramite una istruzione CALL o EXEC). Il comando LABEL richiede infatti di esprimere i parametri x-pos e y-pos in coordinate hardware e quindi di usare SCALEX e SCALEY se si sta operando con coordinate utente.

SCALE (PROGRAMMA/IMMEDIATO)

Consente di scegliere il sistema di coordinate utente più opportuno, definendo la trasformazione tra il sistema di coordinate di default e il sistema di coordinate prescelto.

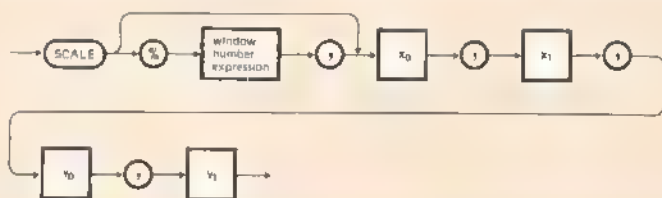



Figura 14-10 Istruzione SCALE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è una espressione numerica, arrotondata all'intero più vicino, il cui valore identifica la finestra su cui operare. Se omesso, il sistema opera sulla finestra attuale.
xØ,x1,yØ,y1	dimensioni della finestra: (in coordinate utente) xØ: ascissa del lato sinistro della finestra (cioè x minimo) x1: ascissa del lato destro della finestra (cioè x massimo)

y∅: ordinata del lato inferiore della finestra (cioè y minimo)

y1: ordinata del lato superiore della finestra (cioè y massimo)

Nota: x1-x∅, y1-y∅ possono essere sia positive che negative, ma non devono mai essere uguali a zero.

Caratteristiche

Dopo che è stata eseguita un'istruzione SCALE, ogni volta che si deve esprimere un valore di una coordinata, detto valore va impostato in coordinate utente.

Il sistema di coordinate è invece quello di default se:

- nessuna istruzione SCALE è stata eseguita, oppure

- è stata eseguita l'istruzione:

  SCALE ∅,511,∅,255

Il sistema di coordinate utente scelto con l'istruzione SCALE resta valido finché:

- non viene eseguita un'altra istruzione SCALE, oppure

- non si esce dall'ambiente BASIC.

Esempi (Video a 4 colori)

SE l'utente imposta...	ALLORA...
COLOR=3,∅,1,5 CR CLS CR LINE(∅,∅)-(511,255) CR	l'istruzione LINE (descritta in seguito) traccia una linea nera su sfondo cyan dal punto di coordinate (∅,∅) al punto di coordinate (511,255). La linea è indicata nella Figura 14-11.

	Questa linea è stata tracciata usando il sistema di coordinate di default, dal momento che nessuna istruzione SCALE è stata eseguita in precedenza.
SCALE -1000,1000,-1000,1000 **CR** LINE (0,0)-(511,255) **CR**	Con l'uso dell'istruzione SCALE viene assunto un nuovo sistema di coordinate. L'esecuzione della medesima istruzione LINE questa volta traccia una linea diversa (vedere la Figura 14-12).

Figura 14-11 Istruzione LINE

Figura 14-12 Istruzioni SCALE e LINE

SCALEX (PROGRAMMA/IMMEDIATO)

Trasforma una coordinata utente nella corrispondente coordinata hardware sull'asse delle x della finestra attuale.

Il valore di ritorno della funzione esprime la misura in pixel del segmento, lungo l'asse x, compreso tra l'origine della finestra (angolo in basso a sinistra) e il punto in esame.

Figura 14-13 Funzione SCALEX

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
coordinate	è la coordinata utente x del punto in esame misurata sull'asse delle x della finestra attuale

SCALEY (PROGRAMMA/IMMEDIATO)

Trasforma una coordinata utente nella corrispondente coordinata hardware sull'asse delle y della finestra attuale.

Il valore di ritorno della funzione esprime la misura in pixel del segmento lungo l'asse y, compreso tra l'origine della finestra (angolo in basso a sinistra) e il punto in esame.

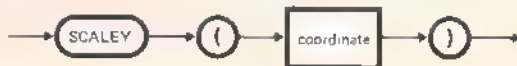



Figura 14-14 Funzione SCALEY

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
coordinate	è la coordinata utente y del punto in esame misurata lungo l'asse delle y della finestra attuale.

VISUALIZZAZIONE DI PUNTI

Visualizzare un punto con un dato colore è la funzione grafica più elementare e l'utente può far questo con le istruzioni PSET e PRESET.

L'istruzione POINT consente invece di conoscere il numero di colore di un pixel specificato.

PSET (PROGRAMMA/IMMEDIATO)

Attiva con il colore specificato o con il colore di foreground (della finestra specificata o della finestra attuale) il pixel di coordinate (x,y) specificate oppure, se si è passati in coordinate utente, il pixel più vicino alle coordinate (x,y).

Figura 14-15 Istruzione PSET

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un' espressione, arrotondata all'intero più vicino, che seleziona la finestra sulla quale PSET deve operare. Se omessa, viene selezionata la finestra attuale.
x,y	sono le coordinate su cui PSET deve operare. Se le coordinate x,y individuano un punto al di fuori della finestra, PSET non attiverà il pixel, a causa del "clipping".

colour	specifica il numero di colore del pixel da visualizzare. Se omesso, il colore usato è quello di foreground.

Nota: Il parametro colour dell'istruzione PSET non fa variare i colori di foreground e background della finestra specificata.

PRESET (PROGRAMMA/IMMEDIATO)

Attiva con il colore di background (della finestra specificata o della finestra attuale) il pixel di coordinate (x,y) specificate, oppure, se si è passati in coordinate utente, il pixel più vicino alle coordinate (x,y).

Figura 14-16 Istruzione PRESET

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica, arrotondata all'intero più vicino, che seleziona la finestra sulla quale PRESET deve operare. Se omessa, viene selezionata la finestra attuale
x,y	sono le coordinate su cui PRESET deve operare. Se x,y individuano un punto al di fuori della finestra, PRESET non attiverà il pixel, a causa del "clipping".

POINT (PROGRAMMA/IMMEDIATO)

Assegna alla variabile alla sinistra del segno di uguale un numero intero che rappresenta il numero di colore del pixel di coordinate (x,y) specificate, oppure del pixel più vicino a dette coordinate, se si è passati in coordinate utente. Le coordinate (x,y) devono cadere all'interno della finestra attuale.

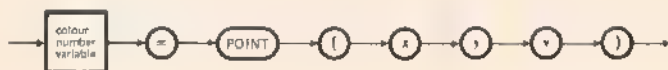



Figura 14-17 Funzione POINT

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
colour number variable	è una variabile, alla quale il sistema assegna un intero che può essere Ø oppure 1 per un sistema bianco e nero, che può variare da Ø a 3 per un sistema a 4 colori e da Ø a 7 per un sistema a 8 colori. Questo intero è il numero di colore del pixel di coordinate (x,y) specificate, o del pixel più vicino a dette coordinate, se si è passati a coordinate utente.
x,y	coordinate del pixel di cui si vuole conoscere il numero di colore.

Esempi

VIDEO	COMMENTI
10 CIRCLE(50,50),20	traccia una circonferenza sul video con centro nel punto di coordinate (50,50) e raggio 20
20 PSET(50,50)	attiva con il colore di foreground il pixel di coordinate (50,50) oppure il pixel più vicino a queste coordinate, se si è passati a coordinate utente
30 A%=POINT(50,50)	assegna un valore intero alla variabile A%: detto valore è il numero di colore del pixel alle coordinate (50,50) o del pixel più vicino a queste coordinate
40 PRINT A%	visualizza il contenuto di A%

VISUALIZZAZIONE DEI CURSORI

Ogni finestra ha due cursori: uno di testo e uno grafico.

Il cursore di testo individua la posizione dove verrà visualizzato il successivo carattere impostato.

La posizione del cursore di testo viene espressa in termini di numero di colonna e numero di riga.

L'utente ha a disposizione la funzione POS (descritta in seguito) per conoscere la posizione del cursore di testo.

Il cursore grafico può essere visualizzato in ogni posizione desiderata. Si noti però che la posizione di questo cursore non varia quando si eseguono istruzioni di grafica.

Con l'uso dell'istruzione CURSOR (descritta in seguito), l'utente può visualizzare o no uno dei cursori, può variarne la forma e stabilirne la frequenza di lampeggio.

La forma standard del cursore grafico è un rettangolo di 2 pixel per 2 pixel.

Il cursore di testo ha, invece, come forma standard un segno di sottolineatura. La visualizzazione di uno dei cursori è possibile solo nella finestra su cui si sta operando; appena l'utente seleziona un'altra finestra, il cursore visualizzato nella finestra precedente scompare, ma viene memorizzato e ogni volta che l'utente ritorna ad operare su quella finestra il cursore riappare con le stesse caratteristiche.

Si noti che quando il cursore di testo viene abilitato, il cursore grafico viene automaticamente disabilitato e viceversa; i due cursori non possono essere visualizzati contemporaneamente.

CURSOR (PROGRAMMA/IMMEDIATO)

Vi sono due formati sintattici diversi per questa istruzione: CURSOR e CURSOR POINT. Specificano, rispettivamente, la posizione e le caratteristiche del cursore di testo e del cursore grafico.

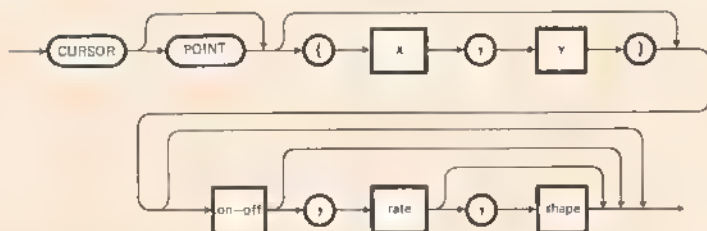



Figura 14-18 Istruzione CURSOR

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
POINT	è una parola chiave opzionale. Viene usata se si vuole operare sul cursore grafico. Se viene omessa, le operazioni vengono eseguite sul cursore di testo

x,y	specificano dove il cursore deve essere posizionato; se si opera sul cursore di testo x e y rappresentano rispettivamente la posizione di colonna e di riga. Se. invece, si opera sul cursore grafico, x e y rappresentano le coordinate di un punto sul video dove il cursore viene posizionato (ed esattamente le coordinate dell'angolo in basso a sinistra dell'area rettangolare rappresentatne la Bit-Map del cursore - vedere parametro "shape").
on-off	stabilisce se visualizzare o no il cursore: Ø= non visualizzare 1= visualizzare
rate	stabilisce se il cursore deve lampeggiare e in caso affermativo la frequenza di lampeggio: Ø= non lampeggia 1-2Ø= frequenza di lampeggi
shape	è un parametro opzionale. Permette di variare la forma del cursore. E' il primo elemento di una matrice intera unidimensionale a sei elementi definita dall'utente. Gli elementi di questa matrice (vedere Tabella 14-2 danno la Bit-Map voluta del cursore che, sia per il cursore di testo che per il cursore grafico, è un rettangolo di 8 pixel (base) per 12 pixel (altezza). Ogni bit della Bit-Map del cursore rappresenta un pixel. Il contenuto della Bit-Map del cursore viene messo in XOR con il contenuto di quella parte della Bit-Map del video che rappresenta l'area occupata dal cursore.

Esempi

SE l'utente imposta...	ALLORA...
CURSOR POINT(8Ø,3Ø): A$=INPUT$(1) CR	il cursore grafico viene posizionato nel punto di coordinate (8Ø,3Ø) L'istruzione A$=INPUT$(1) è stata impostata affinchè il cursore rimanga nella posizione specificata fino a quando l'utente non imposta un carattere da tastiera.
CURSOR POINT(5Ø,5Ø)1: A$=INPUT$(1) CR	il cursore grafico viene posizionato nel punto di coordinate (5Ø,5Ø) e viene visualizzato
CURSOR POINT(5Ø,5Ø)1,1: A$=INPUT$(1) CR	il cursore grafico viene posizionato nel punto di coordinate (5Ø,5Ø), viene visualizzato e ha una frequenza di lampeggio pari ad uno.
CURSOR (32,8)1:A$=INPUT$(1) CR	il cursore di testo viene posizionato alla colonna 32 della riga 8; viene visualizzato e non lampeggia.
CURSOR (32,8)1,Ø,A%(1): A$=INPUT$(1) CR	come sopra, con l'unica differenza che la forma del cursore è stata definita dall'utente (una freccia rivolta verso l'alto)

BIT MAP	ELEMENTO	DECIMALE	ESADECIMALE
ØØØ1ØØØØ ØØ111ØØØ	A%(1)	4152	&H1Ø38
Ø11111ØØ 1111111Ø	A%(2)	31998	&H7CFE
ØØ111ØØØ ØØ111ØØØ	A%(3)	14392	&H3838
ØØ111ØØØ ØØ111ØØØ	A%(4)	14392	&H3838
ØØ111ØØØ ØØ111ØØØ	A%(5)	14392	&H3838
ØØ111ØØØ ØØ111ØØØ	A%(6)	14392	&H3838

Tabella 14-2 La Bit Map del Cursore

Nota: Si ricordi che ogni elemento della matrice intera contiene sedici bit.

POS (PROGRAMMA/IMMEDIATO)

Dà la posizione del cursore di testo nella finestra attuale.

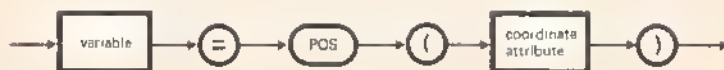



Figura 14-19 Istruzione POS

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	è una variabile numerica, alla quale il sistema assegna un valore intero. Questo valore fornisce o la posizione di riga o la posizione di colonna del cursore di testo (vedere il parametro coordinate attribute qui di seguito)
coordinate attribute	specifica se nella variabile deve ritornare il valore relativo alla posizione di riga o alla posizione di colonna. Ø = posizione di colonna <>Ø = posizione di riga.

COME TRACCIARE FIGURE

La grafica dell'M20 comprende istruzioni che permettono all'utente di tracciare linee e rettangoli (istruzione LINE), circonferenze (istruzione CIRCLE) e figure qualsiasi (istruzione DRAW) e di colorare l'area racchiusa da una figura chiusa (istruzione PAINT).

Traccia una linea o un rettangolo in un determinato colore. L'utente ha la facoltà di colorare la superficie del rettangolo.

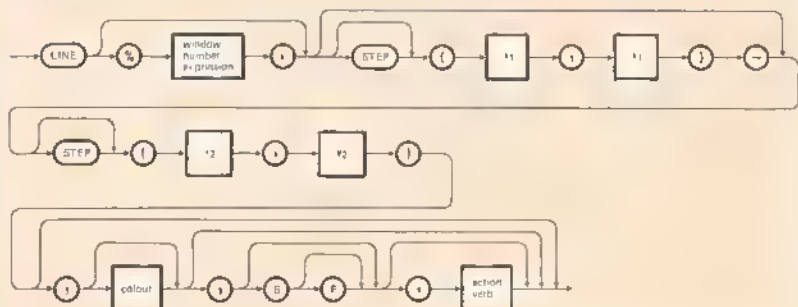



Figura 14-20 Istruzione LINE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica intera, che individua e seleziona la finestra sulla quale l'istruzione LINE deve operare. Se omessa, l'istruzione LINE opera sulla finestra attuale.
STEP	è una parola chiave opzionale. Consente l'uso di coordinate relative. Con l'uso di STEP le coordinate (x_1,y_1) (coordinate iniziali) divengono relative all'ultimo punto tracciato o (in assenza di questo) all'angolo inferiore sinistro della finestra. Le coordinate (x_2,y_2) (coordinate finali) divengono, invece, relative al punto d'inizio della linea (o del rettangolo)

x_1,y_1	sono le coordinate del punto iniziale della linea. Se omesse, la linea inizia dall'ultimo punto tracciato o, in assenza di questo, dall'angolo inferiore sinistro della finestra.
x_2,y_2	sono le coordinate del punto finale della linea
colour	un numero di colore, che individua il colore con il quale deve essere tracciata la linea o il rettangolo. Il valore di default è il colore di foreground della finestra attuale.
B (Box)	è un parametro opzionale, con l'uso del quale è possibile tracciare un rettangolo. Il rettangolo (con i lati paralleli ai bordi della finestra) ha come diagonale, il segmento di coordinate (x_1 ,y_1) e (x_2,y_2)
F (Filled)	è un parametro opzionale, da usare solo con il parametro B. 'BF' traccia un rettangolo, la cui superficie viene colorata con il colore specificato nel parametro colour, o con il colore di foreground.
action verb	è un parametro opzionale, che può assumere i seguenti valori: AND, XOR, OR, NOT, PSET, PRESET. L'opzione PSET fa in modo che la linea o il rettangolo vengano tracciati nel colore indicato. Le opzioni AND, OR e XOR indicano che il colore con cui tracciare la linea o il rettangolo è il risultato della corrispondente operazione logica fra il numero di colore della linea o del rettangolo e il numero di colore dei pixel già presenti su video. Questa operazione viene ripetuta per ogni punto della linea o del rettangolo. L'opzione NOT indica che il colore della linea o del rettangolo è il complemento del numero di colore dei pixel già presenti su video. Con l'operazione PRESET la linea o il rettangolo vengono tracciati con il colore di background della finestra selezionata. Il valore di default del parametro action verb è PSET.

Esempio (Video a 4 colori)

VIDEO	COMMENTI
1Ø COLOR=4,2,4,5 2Ø CLS 3Ø LINE(2Ø6,1ØØ)-(3Ø6,1ØØ) 4Ø LINE(256,2ØØ) 5Ø LINE STEP(-5Ø,-1ØØ) 6Ø PAINT(256,15Ø)	questo programma disegna su un colore di background rosso un triangolo isoscele di colore blu (v. Figura 14-21). L'istruzione PAINT viene descritta nel seguito.

Figura 14-21 Disegno di un triangolo

Nota

Se i parametri specificati per tracciare una linea o un rettangolo sono tali che una parte della linea o del rettangolo cade al di fuori dei confini della finestra, solo la parte interna alla finestra viene visualizzata e la parte esterna viene ignorata ("clipped").

CIRCLE (PROGRAMMA/IMMEDIATO)

Traccia una circonferenza in un dato colore, specificando le coordinate del centro e il raggio.

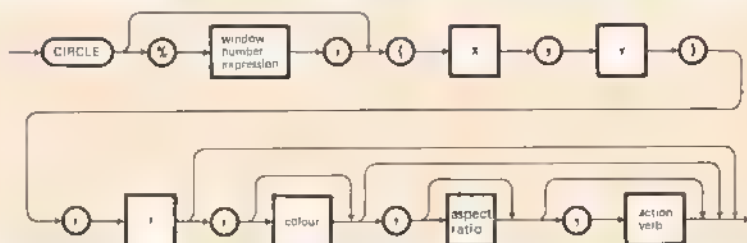



Figura 14-22 Istruzione CIRCLE

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica, arrotondata all'intero più vicino, che seleziona la finestra sulla quale l'istruzione CIRCLE deve operare. Se omessa, viene selezionata la finestra attuale.
x,y	sono le coordinate del centro della circonferenza
r	è il raggio della circonferenza
colour	è un numero di colore che individua il colore con cui disegnare la circonferenza. Il valore di default è il colore di foreground della finestra selezionata

aspect ratio	a causa della diversa densità dei pixel su video lungo l'asse x e l'asse y, l'utente può specificare un valore (numero reale positivo) di questo parametro, al fine di poter tracciare un cerchio collegando all'M20 un video diverso da quello standard. Il parametro è opzionale. Il valore di default è Ø.8Ø7 e dà luogo ad una circonferenza con un video M20 standard.
action verb	è un parametro opzionale. Può assumere uno dei valori seguenti: AND, XOR, OR, NOT, PSET, PRESET. Ognuna di queste opzioni definisce l'operazione da eseguire su ogni punto della circonferenza. L'opzione PSET traccia la circonferenza nel colore indicato. Le operazioni AND, OR e XOR indicano che il colore con cui tracciare la circonferenza è il risultato della corrispondente operazione logica fra il colore indicato o assunto per default e quello dei pixel già presenti su video. L'operazione NOT indica che il colore della circonferenza è il complemento del colore dei pixel già presenti su video. Con l'opzione PRESET la circonferenza viene tracciata con il colore di background della finestra selezionata. Il valore di default del parametro action verb è PSET.

Nota

Quando si usa una istruzione SCALE, il parametro aspect ratio non viene modificato e il parametro r viene determinato in funzione dell'unità di misura sull'asse delle x.

Esempio (Video a 4 colori)

VIDEO	COMMENTI
10 COLOR=2,4,5,0 20 CLS 30 CIRCLE(100,120),90 40 CIRCLE(150,130),120 50 CIRCLE(250,120),100 60 PAINT(180,120)	Il programma disegna tre cerchi che si intersecano. Il colore di fondo è il blu, quello delle circonferenze è il colore di foreground (rosso) e l'area comune ai tre cerchi viene colorata (tramite la PAINT) sempre con il colore di foreground. (Vedere Figura 14-23). L'istruzione PAINT viene descritta nel seguito.

Figura 14-23 Cerchi che si intersecano

Il video può essere pensato come un foglio di carta sul quale poter far scorrere una ipotetica penna chiamata penna virtuale ("virtual pen"). Questa penna può essere spostata dall'utente in qualsiasi posizione del video e, nel far questo, può disegnare (pen down) oppure no (pen up).

L'istruzione DRAW sposta la penna virtuale in qualsiasi posizione all'interno di una finestra e può tracciare linee di un colore specificato.

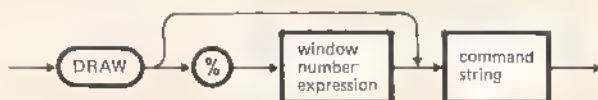



Figura 14-24 Istruzione DRAW

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica intera, che individua e seleziona la finestra, sulla quale DRAW deve operare. Il valore di default è la finestra attuale.
command string	può essere o una variabile stringa o una costante stringa. In entrambi i casi, la stringa è formata da uno o più comandi (elencati nella tabella Comandi); questi comandi controllano il movimento della "penna virtuale". Tutti i comandi, ad eccezione del comando C, possono essere preceduti dall'opzione B, che impedisce di disegnare su video (e quindi ha la funzione di "pen up") e possono essere seguiti da una delle seguenti

	opzioni: AND, XOR, OR, NOT, PSET, PRESET. Ognuna di queste opzioni individua la corrispondente operazione, che viene eseguita su ciascun punto della linea. Queste opzioni vengono specificate con la prima lettera del nome; unica eccezione, l'opzione PRESET, che viene specificata con la lettera R. L'opzione P (cioè PSET) traccia la figura nel colore indicato. Le opzioni A (cioè AND), O (cioè OR), e X (cioè XOR) indicano che il colore della figura è il risultato di un'operazione logica fra il colore indicato e il contenuto precedente del video. L'opzione N (cioè NOT) indica che il colore della figura è il complemento del contenuto del video. L'opzione R (cioè PRESET) traccia la figura nel colore di background. Il valore di default è P.

Nota: I parametri dei comandi (dx,dy,x,y e colour) possono essere espressi come variabili. In questo caso, i nomi delle variabili devono essere scritti fra segni di uguale. Vedere gli esempi qui di seguito.

Comandi

COMANDO	SIGNIFICATO
M dx,dy	sposta la penna dalla posizione attuale (diciamo a,b) alla posizione indicata da (a+dx,b+dy).
J x,y	sposta la penna nella posizione indicata da (x,y).

U dy	sposta la penna in alto di dy posizioni.
D dy	sposta la penna in basso di dy posizioni.
L dx	sposta la penna a sinistra di dx posizioni.
R dx	sposta la penna a destra di dx posizioni.
C colour	stabilisce il colore con cui tracciare le linee. Dopo l'opzione C deve essere specificato un numero di colore. Se l'opzione C non viene specificata, viene assunto l'ultimo colore usato in una precedente istruzione DRAW oppure il colore di foreground della finestra attuale.

Esempi

VIDEO	COMMENTI
9Ø PSET(1Ø,2Ø) 1ØØ X=23 ... 13Ø DRAW "M=X=,25" ...	l'istruzione 9Ø attiva il punto (1Ø,2Ø) con il colore di foreground. L'istruzione 1ØØ pone X=23 L'istruzione 13Ø traccia una linea dalla posizione attuale della penna (1Ø,2Ø) alla posizione (33,45), cioè (1Ø+23,2Ø+25)

```
250 A$="BM 10,2
    D 20 MR 15,-3"
```

l'istruzione 250 assegna alla variabile A$ la seguente stringa di comandi:

- il comando M con l'opzione B per spostare la penna senza disegnare ("pen up") dalla sua posizione attuale (diciamo a,b) alla posizione (a+10,b+2).

- il comando D per spostare la penna in basso di 20 posizioni cioè fino al punto (a+10,b-18)

- il comando M per spostare la penna dalla sua posizione attuale (a+10, b-18) al punto (a+25,b-21).

L'opzione R (PRESET) indica che la linea deve essere tracciata nel colore di background.

```
260 DRAW A$
```

l'istruzione 260 esegue la sequenza dei comandi, specificati nella variabile A$.

Nota

La sequenza di comandi in una istruzione DRAW può essere impostata sia in lettere minuscole che in lettere maiuscole. I comandi possono essere separati l'uno dall'altro da spazi, ma possono anche essere contigui.

Esempio (Sistema a 8 colori)

VIDEO	COMMENTI
1Ø CLS 2Ø DRAW "bjØ,Ø" 3Ø X=511:Y=255 4Ø K=Ø 5Ø FOR I=1 TO 23 6Ø GOSUB 17Ø 7Ø DRAW "u=y=":Y=Y-5 8Ø GOSUB 17Ø 9Ø DRAW "r=x=":X=X-1Ø 10Ø GOSUB 17Ø 11Ø DRAW "d=y=":Y=Y-5 12Ø GOSUB 17Ø 13Ø DRAW "l=x=":X=X-1Ø 14Ø NEXT 15Ø A$=INPUT$(1) 16Ø END 17Ø COL=(K MOD 7)+1 18Ø K=K+1:DRAW "c=col=":RETURN	Il programma disegna una spezzata partendo dal punto di coordinate Ø,Ø (vedi istruzione 2Ø). Detto punto si trova nell'angolo inferiore sinistro del video, dato che non è stata usata un'istruzione SCALE. La subroutine alle linee 17Ø e 18Ø consente di cambiare il colore della linea da disegnare (scegliendo un numero di colore tra 1 e 7 - escludendo quindi il nero, che è il colore di background). Questa subroutine viene attivata ogni volta che viene tracciata una linea. La prima istruzione dalla linea 7Ø traccia una linea verso l'alto di Y unità e la seconda istruzione decrementa il valore di Y di 5 unità. La prima istruzione alla linea 9Ø traccia una linea verso destra di X unità e la seconda istruzione decrementa il valore di X di 1Ø unità. La prima istruzione alla linea 11Ø traccia una linea verso il basso di Y unità e la seconda istruzione decrementa il valore di Y di 5 unità. La prima istruzione alla linea 13Ø traccia una linea verso sinistra di

X unità e la seconda istruzione decrementa il valore di X di 1Ø unità.

Il numero di linee da tracciare viene controllato da un loop con una variabile di controllo I che va da 1 a 23. Questo loop consente quindi di tracciare una spezzata di 23 x 4 = 92 segmenti, ognuno di colore diverso (vedere Figura 14-25).

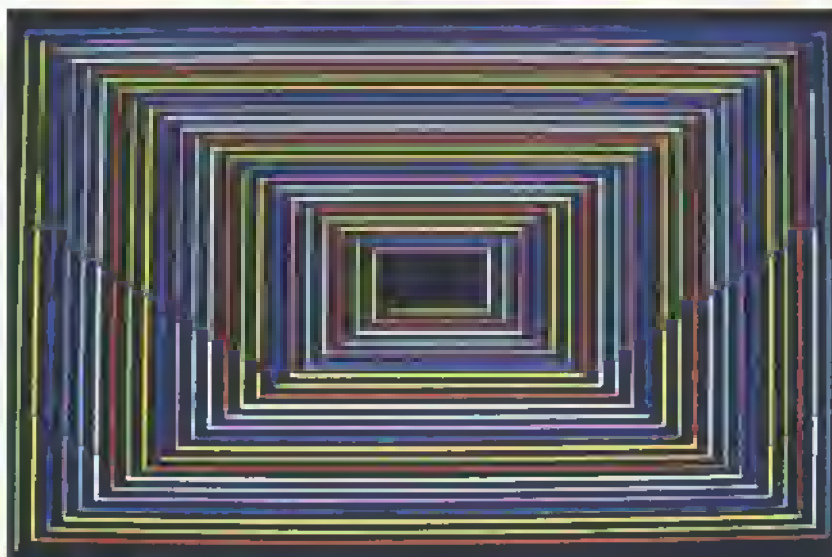



Figura 14-25

Colora l'intera finestra o una sua porzione purché delimitata da un contorno chiuso di un dato colore oppure del colore di foreground. Il sistema inizia a colorare dal pixel di coordinate (x,y) specificate, oppure, se si è passati in coordinate utente, dal pixel più vicino alle coordinate (x,y).

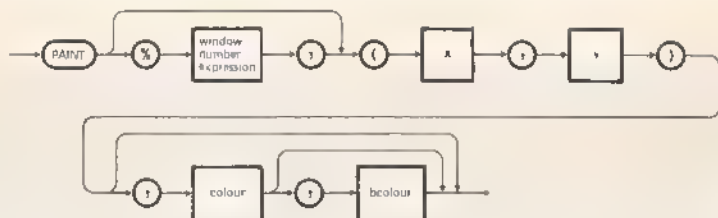



Figura 14-26 Istruzione PAINT

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica il cui valore, arrotondato all'intero più vicino, seleziona la finestra sulla quale l'istruzione PAINT deve operare. Se questo parametro è omesso viene selezionata la finestra attuale.
x,y	sono le coordinate del punto dal quale il sistema inizia a colorare
colour	è un numero di colore che specifica come colorare la finestra o una sua porzione, delimitata da un contorno chiuso. Il valore di default è il colore di foreground

bcolour	è un numero di colore, che specifica il colore del bordo che delimita la figura da colorare. Il valore di default è il colore di foreground.

Nota

Per colorare una figura delimitata da un contorno chiuso, l'utente si assicuri che le coordinate x e y cadano all'interno della figura. Se vengono a trovarsi al di fuori del bordo della figura, il sistema colora soltanto la parte di finestra esterna alla figura medesima.

Per poter utilizzare in modo adeguato l'istruzione PAINT, il bordo della figura deve essere tutto di uno stesso colore (quello specificato dal parametro bcolour, oppure il colore di foreground se detto parametro è omesso). Per esempio, se si vuole colorare una corona circolare, le due circonferenze che la delimitono devono essere dello stesso colore.

Se invece i colori delle due circonferenze sono diversi e:

- il colore specificato dal parametro bcolour è quello della circonferenza esterna, la PAINT colora tutto il cerchio delimitato dalla circonferenza esterna;

- il colore specificato dal parametro bcolour è quello della circonferenza interna, la PAINT colora tutta l'area esterna alla circonferenza interna, fino ai limiti della finestra o fino a incontrare un contorno dello stesso colore della circonferenza interna.

Esempio (Video a 4 colori)

VIDEO	COMMENTI
1Ø COLOR=2,5,4,Ø 2Ø CLS 3Ø CIRCLE(256,128),13Ø,2 4Ø PAINT(256,128),1,2 5Ø LINE(251,123)-STEP(1Ø,1Ø,),3,BF	L'istruzione 1Ø seleziona quattro tra gli otto colori disponibili. L'istruzione 2Ø cancella il contenuto del video e lo colora con il colore di background (in questo caso il blu). L'istruzione 3Ø traccia una circonferenza con un raggio di 13Ø il cui centro è un punto di coordinate (256,128) e colora la circonferenza di rosso. L'istruzione 4Ø colora il cerchio di giallo. L'istruzione 5Ø disegna un rettangolo nero al centro del cerchio (v. Figura 14-27)

Figura 14-27 Disegno di un cerchio con un rettangolo al centro

Esempio (Video a 8 colori)

VIDEO	COMMENTI

<pre>1Ø CLS
2Ø FOR I=1 TO 7
3Ø COLOR I,Ø
4Ø CIRCLE(256,128),11Ø-I*1Ø
5Ø CIRCLE (256,128),11Ø-(I-1)*1Ø
6Ø PAINT (256-11Ø-(I-.5)*1Ø,128)
7Ø NEXT</pre>
	Questo programma traccia 7 corone circolari concentriche. Il centro è nel punto (256,128), cioè coincide con il centro del video.
	
	Il colore di background è il nero (dato che il secondo parametro dell'istruzione COLOR alla linea 3Ø è Ø) e il colore di foreground cambia al variare della variabile di controllo I, da 1 (verde) a 7 (bianco).
	
	Le corone circolari vengono colorate tramite l'istruzione PAINT (alla linea 6Ø). Le coordinate del punto dal quale il sistema inizia a colorare sono state scelte in modo che il punto stesso cada sempre lungo la parallela all'asse x passante per il centro, ed è l'estremo del raggio medio della corona.
	
	Il colore della corona e quello dei cerchi che la delimitano non sono specificati nell'istruzione PAINT, quindi, per default, coincidono con il colore attuale di foreground.
	
	Vedere Figura 14-8

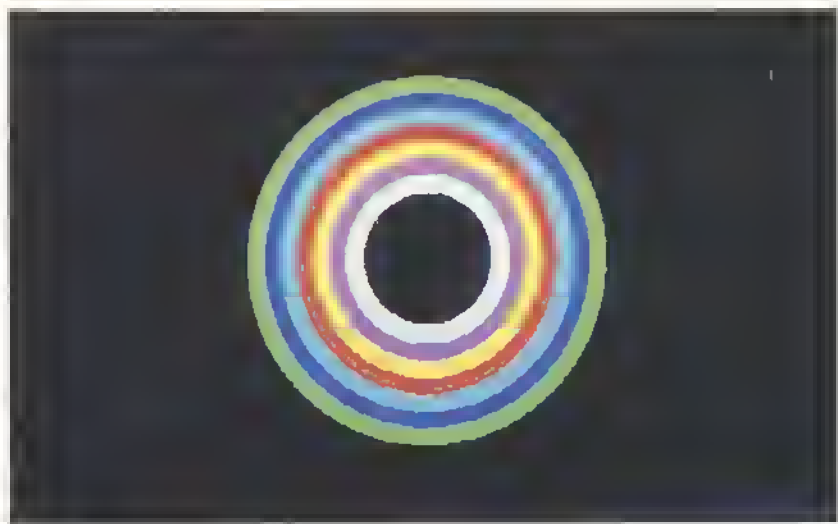



Figura 14-28 Corone concentriche

COME MEMORIZZARE E VISUALIZZARE FINESTRE E RETTANGOLI

L'utente può memorizzare in una matrice intera unidimensionale il contenuto di una finestra o di un'area rettangolare all'interno di una finestra, con l'uso dell'istruzione GET, oppure con l'istruzione PUT può trasferire su video l'immagine di un'area rettangolare, precedentemente memorizzata in una matrice intera unidimensionale.

GET - Grafica (PROGRAMMA/IMMEDIATO)

Memorizza un'intera finestra o un qualsiasi rettangolo all'interno di una finestra in una matrice unidimensionale ad elementi interi (16 bit per elemento).

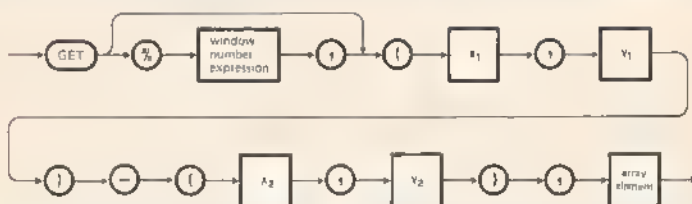



Figura 14-29 Istruzione GET - Grafica

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica, arrotondata all'intero più vicino, che seleziona la finestra sulla quale GET deve operare. Il valore di default è la finestra attuale.
x_1,y_1 x_2,y_2	sono le coordinate che definiscono la diagonale del rettangolo da memorizzare
array element	il primo elemento utilizzabile della matrice unidimensionale, usata nell'istruzione GET. Il sistema utilizza gli elementi della matrice nel modo seguente: - il primo elemento contiene la base del rettangolo; - il secondo contiene l'altezza; - il valore del terzo elemento stabilisce se l'immagine da memorizzare è in bianco e nero o a colori. I successivi elementi della matrice (ciascuno atto a contenere 16 bit) contengono i bit di quella sezione della Bit Map del video che corrisponde al rettangolo considerato.

L'utente deve dimensionare la matrice unidimensionale, usando l'istruzione DIM. La formula seguente permette di calcolare il numero minimo di elementi della matrice:

$$\left(\left(\left\lceil \frac{base}{16} \right\rceil \times altezza\right) \times DT\right)+3$$

dove:

base e altezza sono espresse in pixel

DT = 1 con un video bianco e nero

DT = 2 con un video a 4 colori

DT = 3 con un video a 8 colori

[ ] significa prendere l'intero >= (sempre arrotondato verso l'alto)

PUT - Grafica (PROGRAMMA/IMMEDIATO)

Visualizza un'immagine, precedentemente memorizzata con l'uso dell'istruzione GET in una matrice unidimensionale ad elementi interi (16 bit per elemento).

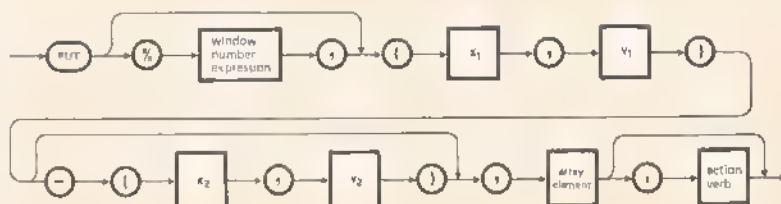



Figura 14-30 Istruzione PUT - Grafica

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
window number expression	è un'espressione numerica intera, che individua e seleziona la finestra sulla quale deve operare l'istruzione PUT. Il valore di default è la finestra attuale.
x_1,y_1 x_2,y_2	sono le coordinate che definiscono la diagonale del rettangolo memorizzato nella matrice unidimensionale, rettangolo il cui contenuto deve essere portato su video. Se la diagonale definita da queste coordinate individua su video un rettangolo di dimensioni diverse da quello memorizzato nella matrice, viene visualizzato il rettangolo minore. Se le coordinate x_2 e y_2 sono omesse, il rettangolo memorizzato viene visualizzato a partire dal punto x_1 e y_1 (vertice superiore a sinistra).
array element	è il primo elemento della matrice unidimensionale, che contiene l'informazione memorizzata con una istruzione GET.
action verb	è un parametro opzionale, che può assumere i seguenti valori: AND, XOR, OR, NOT, PSET, PRESET. Ognuna di queste opzioni definisce l'operazione corrispondente, da eseguire su ogni pixel contenuto all'interno dell'area rettangolare. L'opzione PSET indica che il rettangolo visualizzato è, semplicemente, quello memorizzato nella matrice. Le operazioni AND,OR e XOR stanno invece ad indicare che il rettangolo visualizzato è il risultato di un'operazione logica fra i numeri di colore dei pixel contenuti nella matrice e i numeri di colore dei pixel preesistenti in quell'area rettangolare di video.

L'opzione NOT indica che sarà fatto il complemento dei numeri di colore dei pixel presenti su video. L'opzione PRESET, invece, indica che sarà fatto il complemento dei numeri di colore dei pixel contenuti nella matrice.

Il valore di default del parametro action verb è PSET.

Esempio (Video a 4 colori)

VIDEO	COMMENTI
1 COLOR=2,4,5,Ø 5 DIM 8%(2ØØØ) 1Ø CLS:CIRCLE(256,128),8Ø,3 2Ø LINE(19Ø,6Ø)-(35Ø,195),,BF,XOR 3Ø GET(19Ø,6Ø)-(36Ø,128),8%(Ø) 5Ø CLS:PUT(25Ø,22Ø),B%(Ø)	L'istruzione 5 dimensiona la matrice da usare per memorizzare le scanline del rettangolo. La linea 1Ø cancella quanto contenuto su video, lo colora con il colore di background (blu) e traccia una circonferenza di colore nero. L'istruzione 2Ø traccia un rettangolo sovrapposto al cerchio (vedere la Figura 14-31) e lo colora di rosso. Infatti l'operazione di XOR tra Ø (blu), il numero di colore di background, e 1 (rosso), il numero di colore di foreground, è 1 (rosso). La parte della circonferenza all'interno del rettangolo viene colorata in giallo. Infatti l'operazione di XOR tra il numero di colore 3 e il numero di colore di foreground 1, dà il numero di colore 2. L'istruzione 3Ø memorizza una parte di video nella matrice B%. La linea 5Ø cancella quanto contenuto su video e visualizza l'area rettangolare precedentemente memorizzata (vedere la Figura 14-32).

Figura 14-31 L'immagine su video, dopo l'esecuzione delle istruzioni 1Ø e 2Ø.

Figura 14-32 L'immagine su video, risultante dall'esecuzione dell'istruzione 5Ø.

PRESTAZIONI GRAFICHE FORNITE DAL PCOS

Il PCOS permette all'utente di scegliere la modalità di visualizzazione (256 x 512 o 256 x 480 pixel) tramite il comando SSYS e di preallocare spazio in memoria per un certo numero di finestre tramite il comando SBASIC. (Il PCOS non consente però di aprire, chiudere o selezionare finestre; queste operazioni possono essere fatte solo in BASIC o in ASSEMBLER).

Inoltre i comandi LABEL, SPRINT, LSCREEN, RFONT e WFONT del PCOS, e i parametri +cc e -cc consentono di realizzare alcune importanti funzioni di grafica:

- Il comando LABEL visualizza, con il colore specificato, stringhe di caratteri di dimensione e orientamento variabili nell'ambito dell'intero video o di una data finestra (se richiamato da BASIC);

- Il comando SPRINT riproduce in stampa l'immagine dell'intero video o di una data finestra (se richiamato da BASIC);

- Il comando LSCREEN riproduce in stampa tutti i caratteri presenti sull'intero video o in una finestra (se richiamato da BASIC);

- I comandi RFONT e WFONT (unitamente al Video File Editor) consentono di creare font di caratteri definiti dall'utente;

- I parametri +cc e -cc consentono di visualizzare i primi 32 caratteri della tabella ASCII (codici esadecimali da ØØ a 1F).

Per maggiori dettagli sull'uso di questi comandi e di questi parametri, vedere il "Professional Computer Operating System (PCOS) - Guida Utente".

A. CODICI ASCII

B. EQUIVALENZE DEI CARATTERI IN CODICE ASCII

EQUIVALENZE DEI CARATTERI IN CODICE ASCII

CODICE ASCII		EQUIVALENTE														
DECIMALE	ESADECIMALE	USA	ITALIA	FRANCIA	INGHILTERRA	GERMANIA (1)	GERMANIA (2)	SPAGNA	PORTOGALLO	DANIMARCA	SVEZIA FINLANDIA	NORVEGIA	SVIZZERA FRANCESE	SVIZZERA TEDESCA	GRECIA	IUGOSLAVIA
35	23	(#)	£	£	£	#	#	£	#	£	#	£	£	£	£	#
36	24	($)	$	$	$	$	$	$	$	$	¤	$	$	$	$	¤
64	40	(@)	§	à	@	§	§	§	§	·	@	'	§	§	@	Š
91	5B	[	°	°	[	Ä	Ä	¡	Ã	Æ	Ä	Æ	ä	ä	[	Đ
92	5C	(\)	ç	ç	\	Ö	Ö	Ñ	Ç	Ø	Ö	Ø	ç	ç	\	Ć
93	5D	]	é	§	]	Ü	Ü	¿	Õ	Å	Å	Å	ê	ê	]	Ž
96	60	·	ù	·	·	·	·	·	·	·	·	·	·	·	·	ž
123	7B	{	à	é	{	ä	ä	°	ã	æ	ä	æ	ä	ä	{	đ
124	7C	|	ò	ù	|	ö	ö	ñ	ç	ø	ö	ø	ö	ö	|	ć
125	7D	}	è	è	}	ü	ü	ç	õ	å	å	å	ü	ü	}	ž
126	7E	‾	ì	¨	‾	ß	ß	~	°	¨	‾		è	è	¨	č

* I caratteri compresi in un cerchio sono usati per funzioni in linguaggio BASIC

C. CODICI D'ERRORE E SIGNIFICATO

SOMMARIO

Questa appendice riporta tutti gli errori del BASIC e del PCOS.

In caso d'errore in ambiente BASIC viene visualizzato solo il messaggio senza il codice corrispondente.

In caso d'errore in ambiente PCOS viene visualizzato solo il codice senza il messaggio corrispondente (a meno che il comando EPRINT non sia residente; in questo caso viene visualizzato anche il messaggio).

INDICE

ERRORI PCOS E BASIC

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
1	NEXT without FOR (BASIC)	Una variabile in un'istruzione NEXT non corrisponde ad alcuna variabile in una istruzione FOR eseguita in precedenza.
2	Syntax error (BASIC)	Viene incontrata una linea che contiene alcune sequenze non corrette di caratteri (così come parentesi non accoppiate, parole chiave errate, separatori errati, ecc.).
3	RETURN without GOSUB (BASIC)	Viene incontrata un'istruzione RETURN senza una corrispondente istruzione GOSUB.
4	Out of DATA (BASIC)	Viene eseguita un'istruzione READ quando non ci sono più istruzioni DATA con dati da leggere.
5	Illegal function call (BASIC)	Un argomento fuori del range prescritto viene passato ad una funzione numerica o stringa. Un errore di questo tipo può succedere anche quando: a. un indice di matrici risulti negativo o eccezionalmente grande; b. una funzione LOG abbia un argomento negativo o nullo; c. una funzione SQR abbia un argomento negativo; d. una mantissa negativa abbia un esponente non intero; e. è stato dato un argomento non corretto a una delle seguenti funzioni: MID$, LEFT$, RIGHT$, TAB, SPC, STRING$, SPACE$, INSTR, oppure alla funzione ON...GOTO.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
6	Overflow (BASIC)	Il risultato di un calcolo è troppo grande per essere rappresentato in un numero. Se si verifica un underflow, il risultato viene assunto pari a zero e l'esecuzione continua.
7	Out of memory (PCOS/BASIC)	In BASIC: un programma è troppo grande, ha troppi loop o subroutine, o troppe variabili, o espressioni troppo complesse. In PCOS: è stato richiamato un comando PCOS o una routine Assembler che non può essere allocato/a nella memoria disponibile.
8	Undefined line number (BASIC)	Una istruzione GOTO, GOSUB, ON... GOTO, ON...GOSUB, IF...THEN... ELSE, IF...-GOTO...ELSE, o DELETE fa riferimento a una linea inesistente.
9	Subscript out of range (BASIC)	E' stato indicato un elemento di matrice o con un indice fuori dalle dimensioni della matrice, o con un numero sbagliato di indici.
1Ø	Duplicate definition (BASIC)	Due o più di due istruzioni DIM fanno riferimento alla stessa matrice oppure viene eseguita un'istruzione DIM dopo che per una data matrice si sono assunte le dimensioni di default.
11	Division by zero (BASIC)	Viene incontrata una divisione per zero, oppure si ha un'elevamento a potenza dove il valore della base è zero e l'esponente è negativo. In ogni caso il risultato è l'infinito di macchina: con il segno del numeratore per la divisione per zero, con il segno positivo nel caso dell'elevamento a potenza.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
12	Illegal direct (BASIC)	Viene introdotta una istruzione che non è legale in Modo Immediato.
13	Type mismatch (PCOS/BASIC)	In BASIC: a una variabile numerica viene assegnato un valore stringa; a una funzione che richiede un parametro numerico viene passato un argomento stringa. In PCOS: è stato impostato un valore stringa quando è richiesto un valore numerico e viceversa.
14	Out of string space (BASIC)	Variabili stringa hanno fatto in modo che il BASIC superasse la memoria utente disponibile. Il BASIC alloca lo spazio delle stringhe in modo dinamico, finché non succede un errore di questo tipo.
15	String too long (BASIC)	Si è tentato di creare una stringa con più di 255 caratteri.
16	String formula too complex (BASIC)	Una espressione stringa è troppo lunga o troppo complessa. E' opportuno dividerla in più espressioni semplici.
17	Can't continue (BASIC)	Si è cercato di riprendere l'esecuzione di un programma: 1. che è stato modificato dopo un CTRL C; 2. che non esiste in memoria.
18	Undefined function (BASIC)	Viene richiamata una funzione prima della corrispondente definizione.
19	No RESUME (BASIC)	Il controllo è stato passato a una routine di gestione degli errori che non contiene una istruzione RESUME.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
20	RESUME without error (BASIC)	Viene incontrata una istruzione RESUME prima di entrare in una routine di gestione degli errori.
22	Missing operand (BASIC)	Una espressione contiene un operatore non seguito da operando.
23	Buffer overflow (BASIC)	Si è tentato di introdurre una linea che ha più di 255 caratteri.
26	FOR without NEXT (BASIC)	Si è incontrata un'istruzione FOR senza una corrispondente NEXT.
29	WHILE without WEND (BASIC)	Una istruzione WHILE non ha una corrispondente WEND.
30	WEND without WHILE (BASIC)	Una istruzione WEND non ha una corrispondente WHILE.
31	IEEE Invalid talker/ listener address (BASIC)	Uso illegale di indirizzo di trasmettitore/ricevitore.
32	IEEE: talker = listener address (BASIC)	Si è tentato di trasmettere a un trasmettitore o di ricevere da un ricevitore.
33	IEEE: Unprintable error (BASIC)	Un messaggio d'errore non è stampabile, ossia corrisponde a un codice d'errore non definito.
34	IEEE: Board not present (BASIC)	Si è tentato di usare il package IEEE su una macchina che non ha l'interfaccia IEEE.
35	Window not open (PCOS/BASIC)	Si è cercato di utilizzare una finestra non ancora aperta (può succedere anche in PCOS, quando si esegue un programma Assembler).
36	Unable to create window (PCOS/BASIC)	Le dimensioni della finestra da creare non sono corrette (può succedere anche in PCOS, quando si esegue un programma Assembler).

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
37	Invalid action-verb (BASIC)	Una clausola Action Verb è stata impostata in modo non corretto o non è stata usata nel modo giusto.
38	Parameter out of range (BASIC)	Uno o più parametri sono andati oltre i limiti prescritti.
39	Too many dimensions (BASIC)	Si è tentato di usare una matrice a più dimensioni nella grafica.
5Ø	FIELD overflow (BASIC)	Una istruzione FIELD cerca di allocare più byte di quelli specificati per la lunghezza dei record per un file random.
51	Internal error (BASIC)	Si è verificata una anomalia interna. Segnalare le condizioni d'errore al-l'Organizzazione di Assistenza.
52	Bad file number (BASIC)	Un'istruzione o un comando fa riferimento a un file il cui numero non appartiene all'intervallo specificato all'inizializzazione, oppure il file corrispondente non è aperto.
53	File not found (PCOS/BASIC)	Un comando PCOS o BASIC o una istruzione OPEN fanno riferimento a un file che non esiste sul disco selezionato.
54	Bad file mode (PCOS/BASIC)	<u>In BASIC</u>: si è cercato di usare istruzioni PUT, GET su un file sequenziale, di caricare con LOAD o RUN un file dati o di eseguire una OPEN con un metodo d'accesso diverso da I, O, A o R. <u>In PCOS</u>: può succedere eseguendo un programma Assembler.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
55	File already open (PCOS/BASIC)	In BASIC: è stata eseguita una OPEN su un file già aperto, oppure è stato applicato un comando KILL su un file aperto. In PCOS: può succedere eseguendo un programma Assembler.
57	Disk I/O error (PCOS/BASIC)	Si è verificato un errore di I/O durante un'operazione su disco.
58	File already exists (PCOS/BASIC)	Si è cercato di assegnare a un file un nome identico a uno che esiste già sul disco selezionato.
59	Disk type mismatch (PCOS)	E' stata fatta un'operazione che richiede due dischetti di uguale capacità con due dischi di capacità diversa.
60	Disk not initialised (PCOS)	E' stato fatto un tentativo di accedere a un disco non inizializzato.
61	Disk filled (PCOS/BASIC)	Tutto lo spazio disponibile su disco è già stato usato.
62	End of file (PCOS/BASIC)	Si è verificato un End of File non corretto.
63	Invalid record number (PCOS/BASIC)	Il numero di record è minore o uguale a zero oppure è maggiore di 32768.
64	Invalid file name (PCOS/BASIC)	E' stato usato un nome di file non valido (con troppi caratteri o con caratteri illegali).
66	Direct statement in file (BASIC)	Durante il caricamento di un file programma è stata incontrata un'istruzione immediata (diretta). L'operazione di caricamento (tramite LOAD o RUN) viene interrotta.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
67	Too many files (BASIC)	Si è tentato di creare un nuovo file (con il comando SAVE o l'istruzione OPEN) quando la directory è già satura
69	Volume name not found (PCOS/BASIC)	Il nome del disco specificato non è il nome di un disco in linea.
7Ø	Rename error (PCOS/BASIC)	Si è cercato di attribuire un nome non valido a un disco durante una operazione di rename.
71	Invalid volume number (PCOS/BASIC)	Il numero di drive specificato non è ammesso.
72	Volume not enabled (PCOS/BASIC)	E' stato fatto un tentativo di accedere a un volume non abilitato.
73	Invalid password (PCOS/BASIC)	E' stata specificata una password non valida.
74	Illegal disk change (PCOS/BASIC)	E' stato cambiato il dischetto dall'ultima volta che il file è stato usato.
75	Write protected file (PCOS/BASIC)	E' stato fatto un tentativo di scrivere su un file protetto da scrittura.
76	Error in Parameter (PCOS/BASIC)	Uno o più parametri contengono valori non ammessi.
77	Invalid number of parameters (PCOS/BASIC)	Sono stati indicati troppi parametri.
78	File not OPEN (PCOS/BASIC)	Si è cercato di leggere o scrivere su un file non aperto.
79	Printer error (PCOS/BASIC)	Errore di stampante, come per esempio mancanza di nastro inchiostrato.
8Ø	Copy protected file (PCOS)	E' stato fatto un tentativo di copiare un file che è protetto da copia.

CODICE	DESCRIZIONE (PCOS e/o BASIC)	COMMENTO
81	Paper empty (PCOS/BASIC)	Condizione di fine carta.
82	Printer fault (PCOS/BASIC)	Malfunzionamento della stampante.
92	Command not found (PCOS)	Non è stato trovato il comando specificato sui dischi in linea.
99	Bad load file (PCOS)	La versione del comando non corrisponde alla release di PCOS presente in memoria.
1Ø1	Error in time or date (PCOS)	E' stata specificata un'ora o una data non corrette.
1Ø8	Call-user error (PCOS)	Si è verificato un errore sul richiamo di un comando PCOS o di una routine Assembler.
11Ø	Time-out (PCOS)	Si è verificato un errore di time-out (per esempio si è cercato di usare una stampante non collegata).
111	Invalid device (PCOS)	Il nome del device specificato non esiste.

D. DIFFERENZE TRA RELEASE DEL PCOS

SOMMARIO

Questa appendice indica le differenze tra la release di PCOS 1.3 e la release 2.Ø e successive.

INDICE

DIFFERENZE TRA RELEASE

PCOS RELEASE 1.3	PCOS RELEASE 2.x E SUCCESSIVE
L'hard disk non è supportato.	L'hard disk è supportato (dalla R. 2.x).
I dischetti da 160 Kbyte e da 640 Kbyte non sono supportati.	I dischetti da 160 Kbyte sono supportati dalla R. 2.x e i dischetti da 640 Kbyte dalla R. 3.x.
Il video a 8 colori non è supportato.	Il video a 8 colori è supportato dalla R. 2.x.
I linguaggi ASSEMBLER e PASCAL non sono supportati.	Il linguaggio ASSEMBLER è supportato dalla R. 2.x e il linguaggio PASCAL dalla R. 3.x.
I seguenti comandi PCOS non sono supportati: ASM, BVOLUME, CKEY, DCONFIG, LSCREEN, PDEBUG, PUNLOAD, RFONT SLANG, LINK, VVERIFY, WFONT.	I seguenti comandi PCOS sono supportati (dalla R. 2.x): ASM, BVOLUME, CKEY, DCONFIG, LSCREEN, PDEBUG, PUNLOAD, RFONT, SLANG, LINK, VVERIFY, WFONT. Inoltre sono state aggiunte le seguenti opzioni ai comandi: EDIT (opzione "%g" dalla R. 2.x e "%h" dalla R. 3.x) FLIST (opzione "%h" dalla R. 3.x) PKEY (opzione "%c" dalla R. 3.x) SSYS (opzione "disk time" dalla R. 3.x) Il comando FNEW è stato ampliato per consentire di creare una lista di file (dalla R. 3.x).
Le tastiere Grecia e Iugoslavia e la versione Delta non sono supportate.	Le tastiere Grecia e Iugoslavia sono supportate dalla R. 2.x e la versione Delta dalla R. 3.x.

PCOS RELEASE 1.3	PCOS RELEASE 2.x E SUCCESSIVE
Il comando BASIC è residente.	Il comando BASIC è transiente (dalla R. 2.x).
Il PCOS e il BASIC sono caricati insieme in memoria in fase di inizializzazione.	Solo il PCOS viene caricato in memoria all'inizializzazione (dalla R. 2.x).
All'inizializzazione l'ultimo drive selezionato è sempre il drive Ø.	All'inizializzazione l'ultimo drive selezionato è quello dal quale è stato caricato il PCOS (dalla R. 2.x).
Il prompt del PCOS è >	Il prompt del PCOS è n > , dove n indica l'ultimo drive selezionato (dalla R. 2.x).
Il valore di default del parametro "memory" nell'istruzione CLEAR è 38ØØØ.	Il valore di default del parametro "memory" nell'istruzione CLEAR è 368ØØ (dalla R. 2.x) e 36ØØØ (dalla R. 3.x).
Il comando LABEL del PCOS non ha il parametro "colour".	Il comando LABEL del PCOS ha il parametro opzionale "colour" (dalla R. 2.x).
I valori limite del parametro "position" nell'istruzione WINDOW (per aprire una finestra), se viene fatta una suddivisione orizzontale, è: limite inferiore = valore che indica lo spazio tra due linee contigue di testo + 1 limite superiore = valore che indica l'altezza della finestra genitrice diminuito del limite inferiore + 1	I valori limite del parametro "position" nell'istruzione WINDOW (per aprire una finestra), se viene fatta una suddivisione orizzontale, è: limite inferiore = 1 limite superiore = 255 (a partire dalla R. 2.Ø).

PCOS RELEASE 1.3	PCOS RELEASE 2.x E SUCCESSIVE
Le stampanti PR 1481, PR 2300, PR 430, PR 2835, PR 320 e le macchine per scrivere elettroniche ET 121 ed ET 231 non sono supportate.	Le stampanti PR 1481, PR 2300, PR 430 e le macchine per scrivere elettroniche ET 121 ed ET 231 sono supportate dalla R. 2.x. Le stampanti PR 2835 e PR 320 sono supportate dalla R. 3.x.
Non è possibile visualizzare i primi 32 caratteri della Tabella ASCII.	A partire dalla R. 3.x è possibile abilitare (parametro "+cc") e disabilitare (parametro "-cc") la visualizzazione dei primi 32 caratteri della tabella ASCII, in modo temporaneo (per l'esecuzione di un singolo comando PCOS) o permanente.
Esistono due liste di messaggi di errore, una per il PCOS e una per il BASIC.	La lista dei messaggi d'errore è stata unificata con alcune modifiche, a partire dalla R. 3.x.
Il numero massimo di parametri in un comando PCOS è pari ad 11.	Il numero massimo di parametri in un comando PCOS è pari a 2Ø dalla R. 2.x.

UTILITY GETCONV.BAS

Per assicurare la compatibulità tra release 1.x, 2.x e successive del PCOS, quando un utente ha memorizzato su file delle immagini video ottenute con le istruzioni BASIC GET/PUT della grafica è necessario convertire questi file (sequenziali o random) con l'utility getconv.bas.

Per convertire un file l'utente deve seguire la seguente procedura.

STEP	AZIONE
1	Caricare in memoria il PCOS 2.x
2	Entrare in BASIC impostando: ba **CR** Lanciare l'utility impostando: RUN "getconv.bas" **CR** Nota: è anche possibile impostare direttamente da PCOS ge **CR**
3	Impostare l'identificatore del file da convertire quando appare su video il messaggio di richiesta.
4	Impostare l'identificatore del file di output, quando appare su video il messaggio di richiesta.
5	Impostare "s" se il file da convertire è sequenziale ed "r" se è random.
6	Verificare quanto appare sul video, in particolare la dimensione della matrice unidimensionale a elementi interi che è stata memorizzata su file. Se la dimensione risulta superiore a quella indicata nel programma BASIC sarà necessario variare anche l'istruzione DIM nel programma.
7	Aspettare che la conversione sia finita, cioè che appaia il prompt del BASIC (Ok).

E. ISTRUZIONI, COMANDI E FUNZIONI BASIC

SOMMARIO

Questa appendice riporta, in ordine alfabetico, le istruzioni, i comandi e le funzioni BASIC, indicando la pagina (o le pagine) del manuale dove l'argomento viene illustrato in dettaglio.

Se una istruzione, una funzione o un comando può essere usato sia in una linea di programmazione sia in una linea ad esecuzione immediata, viene specificato PROGRAMMA/IMMEDIATO; se può essere usato solo in una linea ad esecuzione immediata, viene specificato IMMEDIATO; se può essere usato solo in una linea di programma, viene specificato PROGRAMMA.

INDICE